creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학석사 학위논문

Sub-pixel index를 활용한 compound 영상의 텍스트 블록 압축

Text block compression of a compound image using a sub-pixel index

2017 년 8 월

서울대학교 대학원

전기 정보 공학부

박 찬 희

Sub-pixel index를 활용한 compound 영상의 텍스트 블록 압축

Text block compression of a compound image using a sub-pixel index

지도교수 이 혁 재

이 논문을 공학석사 학위논문으로 제출함
2017 년 7 월

서울대학교 대학원
전기 정보 공학부
박 찬 희

박찬희의 공학석사 학위논문을 인준함
2017 년 7 월

위 원 장 _____ (인)

부위원장 _____ (인)

위 원 _____ (인)

초 록

본 논문에서는 compound image의 텍스트 블록을 압축하기 위한 알고리즘을 제안하였다.

기존의 텍스트 블록을 압축하는 대표적인 알고리즘인 SPGC 알고리즘은 텍스트 블록에서 나타나는 sub-pixel의 선형적인 특징을 gradient로 코딩하는 방식이다. 하지만 SPGC 알고리즘은 높은 압축률과 PSNR을 갖지만 텍스트 블록에서 나타나는 gradient의 반복적인 특징을 제대로 담아내지 못하였다.

따라서 본 논문에서는 gradient의 반복적인 특징을 고려하여 효율적으로 gradient를 압축할 수 있는 방법을 제안하였다. 본 논문에서 제안하는 알고리즘은 global index compression과 local index compression으로 이루어져 있다. 두 알고리즘 모두 텍스트 영상에서 반복적으로 나타나는 gradient나 gradient로 이루어진 pattern을 dictionary에 저장하여 dictionary의 index로 코딩하는 dictionary index 방식으로 구현하였다.

Global index compression은 텍스트 블록 전체 영역에서 반복적으로 나타날 가능성이 있는 gradient에 대해 index로 코딩하는 방식이다. Global index의 dictionary는 특정 조건을 만족하는 gradient만을 entry로 갖는다. Dictionary에 저장된 gradient에 대해서 variable length code 방식으로 index bit를 부여하여 압축 효율을 증가시켰다. 또한 dictionary의 최대 index bit는 각각의 블록 내에 존재하는 gradient의 분포에 따라 adaptive하게 결정된다.

Local index compression은 global index compression 방식으로 압축되지 않는 gradient에 대해 적용하기 위한 알고리즘이다. Global index compression 방식으로 압축되지 않는 gradients는 complex pattern의 형태를 띄게 되며, local index dictionary는 이러한 complex

pattern을 entry로 갖는다.

본 논문에서는 이러한 2가지 알고리즘을 이용하여 텍스트 블록의 sub-pixel 영역에서 나타나는 gradient들에 대해 index 방식으로 압축하였다.

본 논문에서 제안하는 알고리즘은 기존의 SPGC 방식과 비교해서 20~25% 높은 압축 효율을 보였으며, PSNR 측면에서는 1~1.5dB의 성능 향상을 확인할 수 있었다.

주요어 : 복합 이미지, 압축, 텍스트 블록, 사전 방식, 기울기, 적응 인덱스 부여
학 번 : 2015-22787

목 차

표 목차

그림 목차

제 1 장 서 론

1.1 연구의 배경

최근 네트워크 속도의 발달로 인해 클라우드 컴퓨팅, N-Screen, VDI 와 같은 컴퓨터 가상화 시스템에 대한 관심이 높아지고 있다. 이와 관련하여 컴퓨터 스크린으로 표현되는 compound image의 전송이 중요한 과제로 떠오르고 있다.

컴퓨터 가상화 시스템은 사용자로 하여 장소에 상관없이 원하는 데이터를 사용할 수 있도록 도와준다. 기존의 1:N의 다수에게 전달하는 전송 방식과는 다르게 클라이언트와 서버 간의 1:1 이나 1:N의 상호 정보 전송 방식을 통하여 전송한다. 이러한 상호간의 데이터를 전달하는 전달 방식에서는 실시간 영상 전송이 중요한 고려사항이 된다.

이때 생성되는 영상은 다양한 프로그램으로 생성되는 영상이기 때문에 동영상, 사진 프로그램 등으로 생성된 이미지 영상이나 웹, 문서프로그램 등으로 생성된 텍스트 영상 등 다양한 특성을 가진 영상의 조합으로 이루어진다. 이러한 다양한 특성을 가진 영상을 실시간으로 전송하기 위해서 영상을 다양한 방식으로 압축하여 클라이언트에게 전송하는데 이때 서버와 클라이언트에 다양한 압축 방식들이 존재하여야 한다. 이는 곧 클라이언트의 성능도 서버와 비슷한 수준을 갖추도록 요구 받는다. 하지만 1:N의 상호 정보 전송 시스템의 높은 서버 성능에 모든 클라이언트의 성능을 맞추는 것은 현실적으로 불가능하며 이는 곧 가상화 시스템의 확산을 저해하는 요인으로 작용한다.

따라서 최근에는 이러한 문제점을 해결하기 위해 클라이언트의

성능에 제약을 받지 않는 thin client 구현이 주목받고 있다. Thin client를 위하여 영상을 생성하는 상위 프로그램과 상관없이 디스플레이 되는 영상 자체를 캡쳐–및 압축하여 전송하는 방법이 많은 관심을 받고 있다. 이러한 방식은 클라이언트가 영상을 생성하는 프로그램에 영향을 받지 않기 때문에 클라이언트의 성능에 대한 요구를 크게 줄일 수 있다. 하지만 영상의 크기가 점점 증가하기 때문에 compound image의 크기를 효과적으로 줄일 수 있는 압축 방식이 중요하다. 또한 compound image는 앞에서 설명한 것과 같이 다양한 특성을 갖고 있기 때문에 각 영상의 특성에 맞는 압축 알고리즘을 사용하여 압축하여야 한다. 이에 따라 기존의 한가지 압축 알고리즘으로 영상 전체를 압축하던 방식에서 각각의 영상의 특성에 따라 압축 알고리즘을 구분하여 사용하는 방법이 연구되고 있다. 이를 위해 영상의 특성을 효과적으로 나누는 방법과 각각의 영역에 적합한 압축 방식을 사용하는 것이 중요하다. 영상의 특성을 나누는 방식은 크게 layer 기반의 접근과 블록 기반의 접근 방식으로 나누어 진다.

Layer 기반 접근 방식은 대표적으로 MRC(Mixed Roster Content)가 있으며 각 layer의 특성에 따라 background, foreground, mask layer로 영상을 나누어 각각의 layer에 맞는 압축 알고리즘을 사용한다[1-6]. Layer 기반 접근 방식은 복잡한 영상을 layer로 나누고 각 layer의 특성에 맞는 알고리즘을 사용하기 때문에 높은 압축 효율과 화질을 보장하지만 각각의 픽셀을 분류하여 layer를 구분하기 때문에 복잡도가 높아진다는 단점이 있다.

블록 기반 접근 방식은 영상을 16*16 이나 32*32 와 같은 non-overlapping 블록으로 나누고 각각의 블록에 적합한 압축 방식을

사용하여 압축을 진행한다.

Lin의 SPEC 논문[7]의 경우 블록을 color count를 사용하여 text, graphic, picture 블록으로 구분한다. 이렇게 구분된 블록들을 각 특성에 맞는 알고리즘을 선택하여 압축한다. 가독성과 화질에 민감한 text와 graphic 블록에는 lossless 압축 방식인 LZW를 사용하고, 화질에 덜 민감한 picture의 경우에는 JPEG를 사용하여 압축하고 있다. 하지만 텍스트 영역에 사용하는 LZW의 경우 계산량이 많아 복잡도가 높아지기 때문에 수행 시간을 증가시키는 문제점을 가지게 된다.

Lan의 논문[8]의 경우 텍스트 블록에서 복잡한 계산량을 줄이기 위해 블록 기반의 BCIM(Base Color and Index Map)을 사용하여 텍스트 블록을 압축한다. 텍스트 블록의 다양한 컬러에 대하여 최적의 base 컬러를 선택하여 index 방식으로 압축하기 때문에 기존의 무 손실 기반의 텍스트 블록 압축 방식에서 압축률의 효율성을 위한 압축 방식으로 판단된다. 하지만 base 컬러를 선택하는 과정에서 복잡한 알고리즘을 선택함으로써 여전히 높은 복잡도를 갖게 된다.

이러한 문제점을 극복하기 위해 Pan의 논문[9]에서는 base 컬러를 선택하는 과정에서 histogram 방식을 사용하여 복잡도를 개선하였다. 하지만 [10]에서 설명하는 텍스트 영상의 complex rendered image의 경우 simple rendered image와 달리 텍스트가 다양한 컬러 값을 갖기 때문에 최적의 base 컬러를 선택하더라도 제한된 수의 base 컬러 선택은 화질 저하를 가져온다.

Kim의 논문[10]에서는 영상의 sub-pixel gradient를 이용하여 블록을 구분한다. 텍스트와 이미지 블록은 sub-pixel 영역에서 서로 다른 gradient 특성을 가지고 있기 때문에 이를 통하여 각각의 블록을

구분한다. 또한 텍스트 영상을 압축하기 위해서 sub-pixel gradient를 사용하는 SPGC 압축 방식을 사용한다. SPGC 방식은 텍스트 블록에서 sub-pixel gradient의 선형적 특성을 이용하여 압축률을 크게 개선하였다.

1.2 연구의 내용

본 논문에서는 새로운 방식의 텍스트 블록 압축 알고리즘을 제안한다. 제안하는 알고리즘은 [10]에서 제안하는 SPGC 알고리즘을 기반으로 한다. SPGC 알고리즘은 텍스트 블록의 sub-pixel 영역에서 나타나는 gradient의 선형성을 이용함으로써 높은 압축률과 SPNR을 가질 수 있었다. 하지만 텍스트 블록에서 나타나는 gradient들의 여러가지 특성을 알고리즘에 반영하지 못하였다. 실제로 텍스트 블록의 sub-pixel 영역에서는 서로 유사한 gradient가 반복적으로 나타난다. 또한 각각의 row에서 여러 개의 gradient가 모여서 특정한 패턴을 반복적으로 만든다. 따라서 본 논문에서는 이러한 텍스트 블록의 2가지 특성을 고려한 텍스트 블록 알고리즘을 제안한다. 제안하는 알고리즘은 텍스트 블록 전 영역에 걸쳐서 반복적으로 나타나는 gradient의 특성을 고려한 global index 코딩과 row단위로 나타나는 gradient의 패턴을 고려한 local index 코딩을 수행한다. 두가지 index 코딩을 위하여 dictionary를 사용한다. Dictionary는 보통 Lampel Ziv[11] 와 같은 lossless 압축 방식에 사용하는 것이 일반적이지만 텍스트 블록의 경우 화질에서 손실을 보더라도 압축률과 복잡도를 낮추기 위해 손실 압축 방식들을 많이 사용하기 때문에 논문에서는 lossy 압축 방식에

적용하기 위한 dictionary 방식을 제안한다. dictionary 방식은 dictionary 생성 및 검색 과정에서 시간적인 한계를 갖는다. 또한 dictionary의 size가 커지면 커질수록 index로 사용되는 bit의 크기 또한 커질 수 밖에 없으며 이는 곧 압축률의 감소로 이어진다. 이러한 dictionary 방식의 한계를 극복하기 위하여 본 논문에서는 텍스트 블록의 모든 gradient를 dictionary에 저장하지 않고 특정한 조건에 맞는 gradient만을 dictionary에 저장한다. 또한 index로 사용되는 bit의 수를 줄이기 위해 dictionary에 있는 모든 gradient을 index로 사용하지 않고 블록 내에서 빈번하게 발생하는 gradient에 대해서만 index 코딩을 수행한다. 이때 각각의 텍스트 블록에서 빈번하게 발생하는 gradient의 개수가 서로 다르기 때문에 각각의 블록마다 서로 다른 index 코딩 bit를 갖게 되며, 이러한 index bit는 압축 효율을 최대화 할 수 있는 방향으로 선택된다.

1.3 논문의 구성

본 논문은 다음과 같이 구성되어 있다. 2장에서는 본 논문의 이해를 돕기 위해 기존의 sub-pixel 영역에서 텍스트 블록을 압축하는 SPGC 방식에 대하여 소개한다. 3, 4장에서는 본 논문에서 제안하는 global index 압축 방식과 local index 압축 방식에 대해 설명한다. 3장에서는 텍스트 블록에서 나타나는 gradient의 반복적인 특성을 고려한 global index 압축 방식에 대해 설명하고 4장에서는 텍스트 블록에서 row단위로 나타나는 gradient의 패턴을 고려한 local index 방식에 대해 설명한다. 마지막으로 5장에서는 본 논문에 대한 결론을 맺는다.

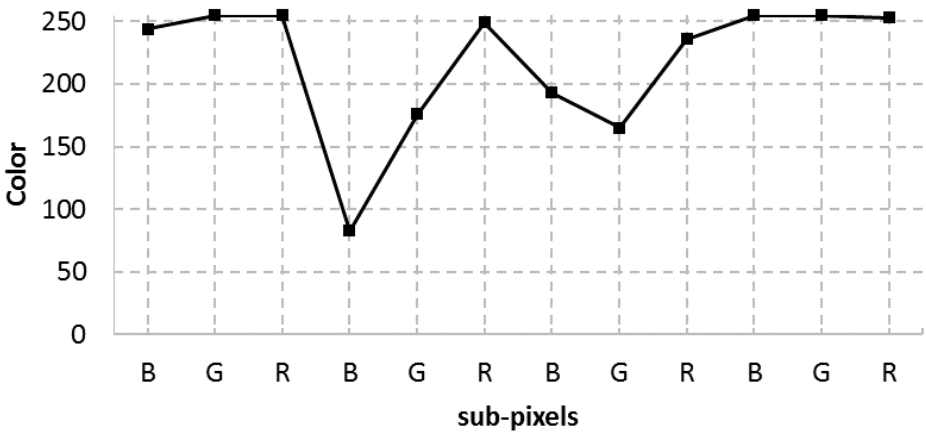
제 2 장 기존의 Sub-pixel gradient 코딩

본 장에서는 텍스트 블록을 압축하기 위해 [10]에서 제안하는 SPGC(Sub-Pixel Gradient Compression)에 대하여 설명한다. SPGC 방식은 텍스트 블록에서 픽셀을 구성하는 R, G, B sub-pixel을 각각 따로 구분하여 압축하는 기존의 방식과 달리 R, G, B sub-pixel을 구분하지 않고 순서대로 나열하여 압축을 진행한다. De-colorization과정을 통해 sub-pixel은 증가 또는 감소하는 경향성을 갖게 되며, SPGC 방식은 이러한 경향성을 갖는 gradient를 이용하여 압축하는 알고리즘 이다.

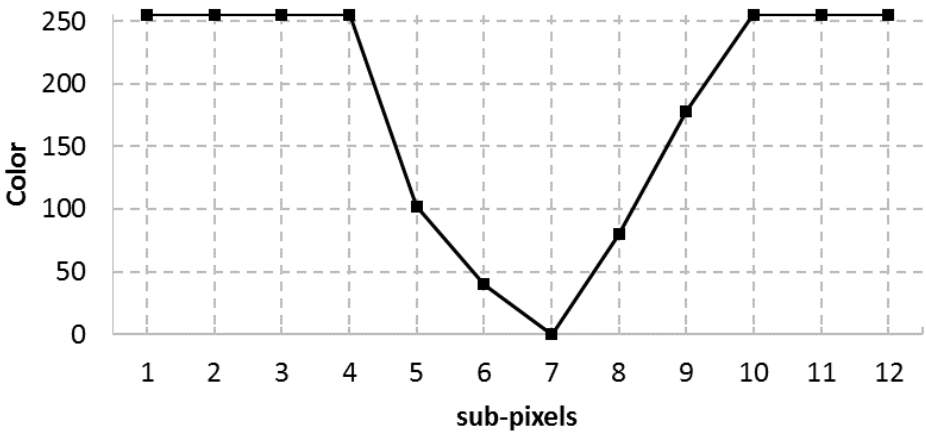
2.1 De-colorization

일반적인 텍스트 영상들은 Alpha blending[12] 과정으로 colorized rendered 되어 있으므로 gradient의 경향성을 확인하기 힘들다. 하지만 de-colorization을 통하여 글자의 경계에서 gradient의 증가 또는 감소하는 경향성을 복원할 수 있다[10]. 그림 (2.1)의 (a)는 colorized rendered 되어 있는 텍스트의 B, G, R 컬러 값을 그래프로 나타낸 것이다. (b)는 de-colorization을 통하여 배경색을 흰색(255)으로, 텍스트의 컬러를 검은색(0)으로 각각 변환하였을 때 컬러 값을 보여준다. (a)의 경우 sub-pixel의 컬러 값들이 어떠한 경향성도 갖지 않는다. 하지만 (b)의 경우 글자의 경계에서 감소 또는 증가하는 방향으로 일정한 gradient의 경향성을 갖는 것을 확인할 수 있다.

따라서 sub-pixel gradient를 활용하기 위해서는 de-colorization을 통한 gradient의 경향성 복원이 필수적이다.



(a) Colorized text



(b) De-colorized text

그림 2.1 De-colorization 변환 전과 후의 sub-pixel 컬러 값

2.2 텍스트 블록 코딩 방법

SPGC 방식은 텍스트 경계에서 발생하는 gradient인 부분과 gradient가 아닌 부분으로 나눠서 압축을 진행한다. Gradient인 부분은 gradient가 직선이라는 가정하에 gradient를 복원하기 위한 최소 정보를 저장하여 SPGC(Sub-Pixel Gradient Coding) 방식으로 압축하고, gradient가 아닌 부분은 연속하는 3개의 sub-pixel 컬러를 확인하여 모두 같은 컬러 값일 경우 RGB를 한번에 whole-pixel coding 하고 다를 경우 RGB를 각각 sub-pixel 단위로 non-gradient coding 한다. SPGC 방법의 전체적인 흐름도는 그림(2.2)와 같다.

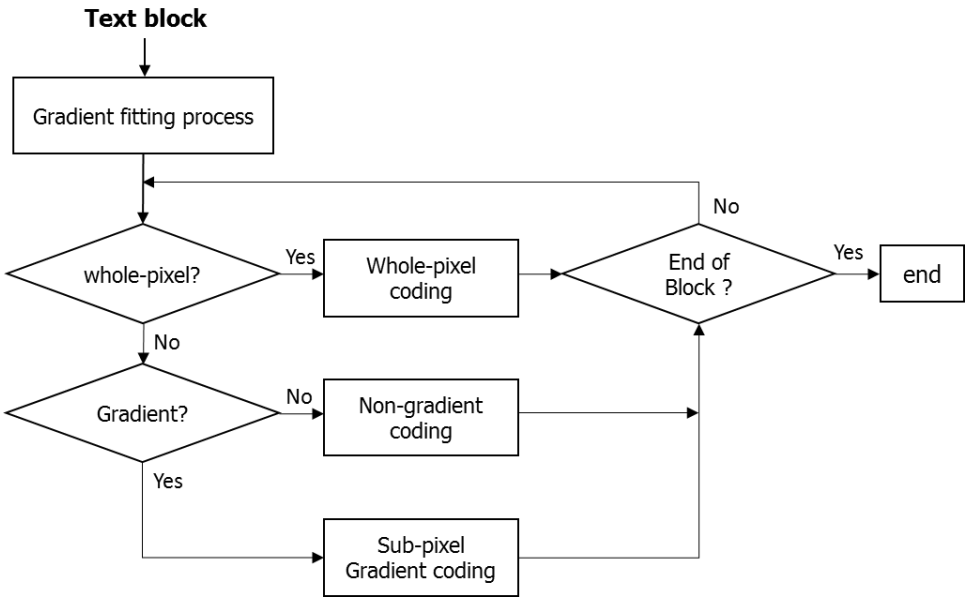
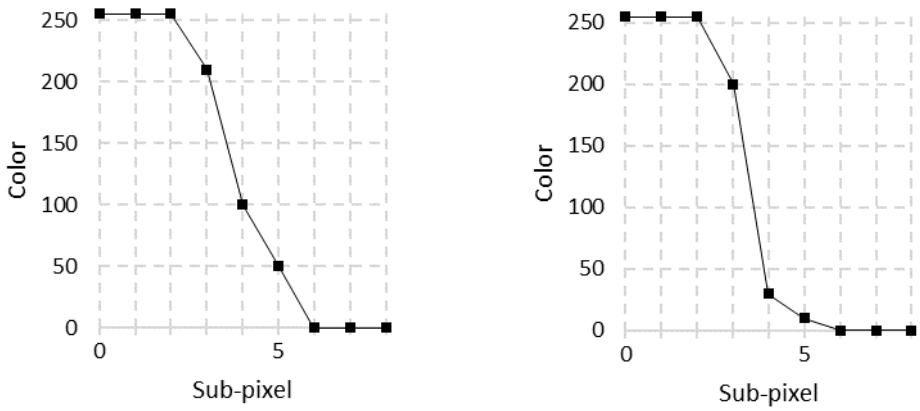


그림 2.2 SPGC 방법

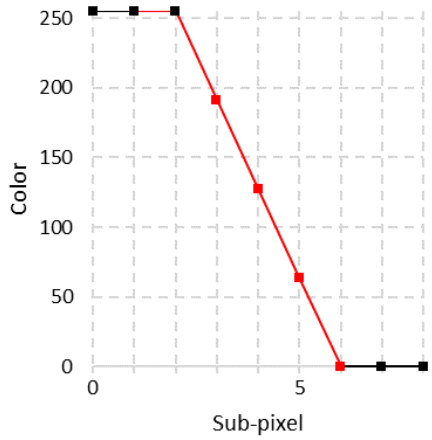
2.2.1 Gradient 부분 코딩 방법

Gradient를 코딩할 경우에는 기본적으로 sub-pixel 영역에서 gradient의 선형적인 특성을 활용하여 코딩을 진행한다. Encoding 과정에서는 gradient를 구성하는 sub-pixel의 정보를 비트스트림에 저장한다. 이때 저장되는 sub-pixel의 정보로는 1) gradient를 구성하는 sub-pixel의 개수, 2) gradient의 방향, 3) gradient가 끝나는 지점에서의 컬러 값으로 총 3가지 정보를 저장하게 되며, 반대로 decoding 과정은 비트스트림에 저장된 sub-pixel의 정보를 통해 gradient의 선형적인 특성을 복원한다.

이때 서로 다른 gradient가 decoding 과정에서 동일한 gradient로 복원될 수 있다. 그림(2.3)은 서로 다른 2개의 gradient가 서로 같은 gradient로 복원된 모습을 나타낸다. (a)의 gradient 2개는 서로 다르지만 gradient를 구성하는 sub-pixel의 개수와 gradient의 방향, 끝점의 컬러 값이 모두 같다. 따라서 SPGC를 통해 gradient를 압축하고 복원하면 (b)와 같이 하나의 gradient로 복원된다.



(a) 압축하기 전의 서로 다른 2개의 gradient



(b) SPGC를 통해 복원된 gradient

그림 2.3 동일한 gradient로 복원되는 서로 다른 gradient

2.2.2 Gradient가 없는 부분 코딩 방법

Gradient가 없는 부분의 코딩은 크게 whole-pixel 단위 코딩과 sub-pixel 단위 코딩으로 나눠진다. Sub-pixel 단위의 압축 방법은 sub-pixel에서 gradient를 압축하는 경우에 적은 비트스트림으로 gradient를 이루는 모든 sub-pixel을 코딩할 수 있다는 점에서 효율적이지만 gradient가 아닌 배경부분에서는 컬러 값의 변화가 없는 일정한 패턴을 갖기 때문에 sub-pixel 단위로 압축을 하게 될 경우 불필요한 bit 수를 늘리게 된다. 따라서 whole-pixel 단위의 코딩에서는 RGB 3개 channel의 데이터를 한꺼번에 코딩하도록 한다. 이 경우 RGB의 픽셀 값이 모두 동일하고 바로 이전 sub-pixel 값과 픽셀 값이 동일 할 경우 이를 1bit로 할당하여 한다. 하지만 텍스트가 나타나는 부분에서 RGB의 픽셀 값이 바로 이전 sub-pixel 컬러 값과 모두 같지 않고 1~2개만

같을 경우에는 sub-pixel 단위의 코딩 방식으로 전환하여 각각의 R, G, B의 값을 sub-pixel 단위로 코딩하도록 한다. 예를 들어 R, G만 바로 이전의 sub-pixel 컬러 값과 같고 B는 다를 경우 R, G는 각각 1bit씩 할당하여 코딩하고 B 부터는 gradient의 시작을 의미하므로 앞에서 설명한 gradient 부분의 코딩 방식으로 코딩하도록 한다.

제 3 장 Global index 코딩

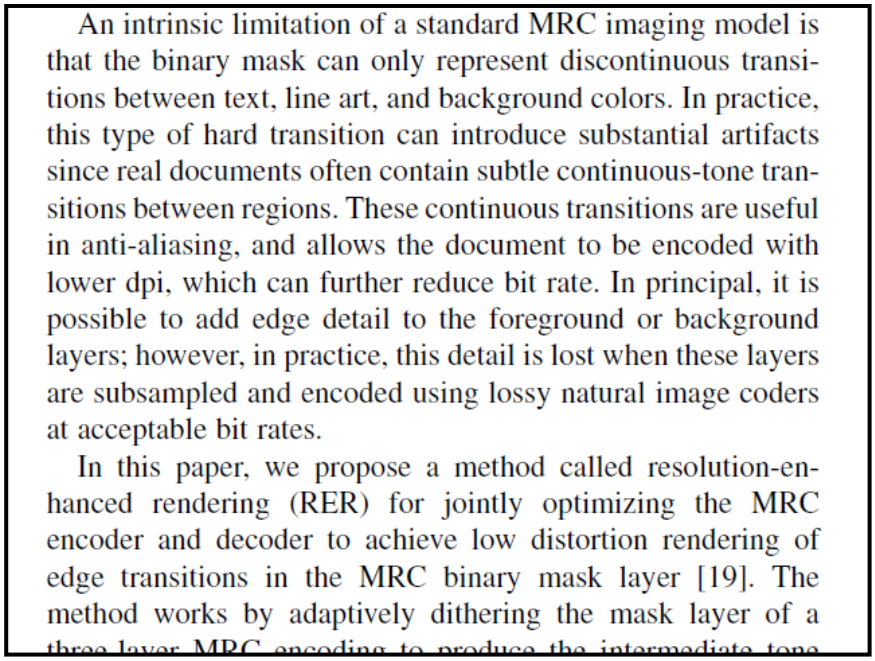
Global index 코딩은 텍스트 블록의 전 영역에 걸쳐 나타나는 gradient의 반복적인 특징을 반영한 코딩 방법이다. 본 장에서는 텍스트 블록에서 나타나는 gradient의 특징과 global index 코딩 방법에 대해서 설명한다.

3.1 텍스트 블록의 gradient 특징

텍스트 블록의 sub-pixel gradient는 [10]에서 제안한 de-colorization 과정을 통해 gradient의 선형적인 특징을 복원할 수 있었다. 또한 이와 더불어 de-colorization을 통해 복원된 대부분의 gradient는 시작 지점과 끝 지점이 각각 배경 컬러와 텍스트 컬러로 변환하게 된다.

그림(3.1)은 640*480 크기의 complex rendered 텍스트[10] 영상이며, 표(3.1)은 텍스트 영상에서 나타나는 gradient들의 중복 횟수가 가장 많은 상위 12개를 나열한 것이다. 그림(3.1)의 텍스트 영상에서 추출한 gradient의 총 개수는 35,656개이며, 중복되는 것을 제외한 gradient의 종류는 1,859개이다. 이 중 상위 12개 gradient의 중복 횟수가 12,415개로 이는 총 gradient 개수의 35%에 해당하며, 하위 1200개의 gradient는 중복 횟수가 10개 이하로 나타난다. 이를 통해 텍스트 영상에서 추출한 1,859가지 종류의 gradient 중 소수의 특정한 gradient가 텍스트 영상의 전 영역에서 반복되어 나타나는 것을

알 수 있다.


An intrinsic limitation of a standard MRC imaging model is that the binary mask can only represent discontinuous transitions between text, line art, and background colors. In practice, this type of hard transition can introduce substantial artifacts since real documents often contain subtle continuous-tone transitions between regions. These continuous transitions are useful in anti-aliasing, and allows the document to be encoded with lower dpi, which can further reduce bit rate. In principal, it is possible to add edge detail to the foreground or background layers; however, in practice, this detail is lost when these layers are subsampled and encoded using lossy natural image coders at acceptable bit rates.

In this paper, we propose a method called resolution-enhanced rendering (RER) for jointly optimizing the MRC encoder and decoder to achieve low distortion rendering of edge transitions in the MRC binary mask layer [19]. The method works by adaptively dithering the mask layer of a three layer MRC encoding to produce the intermediate tone

그림 3.1 Complex rendered 텍스트 (PDF, 640 * 480)

또한 표(3.1)에서 rank1과 rank2의 gradient는 서로 방향만 반대이고 sub-pixel의 값이 같은 gradient라는 것을 확인할 수 있으며, 나머지 10개의 gradient도 서로 짝을 이룬다는 것을 알 수 있다. 결국 gradient의 방향을 구분하지 않으면 표(3.2)와 같이 6개의 gradient가 전체 gradient의 35%를 차지한다는 것을 알 수 있다.

표 3.1 [그림 3.1]영상에서 중복 횟수 상위 12개의 gradient

Rank	sub-pixel value	counts
1	0, 70, 157, 217, 255	1,283
2	255, 217,157, 70, 0	1,263
3	0, 51, 118, 189, 235, 255	1,186
4	255, 217,157,85,0	1,145
5	0, 25, 70, 157, 215, 255	1,141
6	255, 235, 189, 118, 51, 0	1,002
7	0, 85, 157, 217, 255	979
8	0, 25, 118, 189, 255	956
9	255, 215, 157, 70, 25, 0	952
10	0, 25, 121, 189, 255	909
11	255, 189, 121, 25, 0	870
12	255, 189, 118, 25, 0	729
Sum		12,415

표 3.2 [표 3.1]결과에서 gradient의 방향을 구분하지 않은 경우

Rank	sub-pixel value	counts
1	0, 70, 157, 217, 255	2,546
2	0, 51, 118, 189, 235, 255	2,188
3	0, 85, 157, 217, 255	2,124
4	0, 25, 70, 157, 215, 255	2,093
5	0, 25, 121, 189, 255	1,779
6	0, 25, 118, 189, 255	1,685
Sum		12,415

SPGC 코딩 방식에서 gradient를 코딩할 때, gradient의 처음 시작 컬러 값과 끝 컬러 값만을 이용하여 gradient의 정보를 저장한다. 이때 gradient의 시작과 끝점의 컬러 값과 gradient를 이루는 sub-pixel의 개수가 같으면 sub-pixel의 컬러 값과 관계없이 decoding 과정에서 동일한 gradient로 복원된다. 따라서, 표(3.2)에서 rank1, rank3, rank5, rank6은 gradient를 이루는 sub-pixel의 개수가 총 5개 이고 처음 시작과 끝점의 컬러 값이 0 과 255 이므로 decoding 과정에서 같은 gradient로 복원된다. rank2 와 rank4 gradient 또한 sub-pixel의 개수가 6개로 같으므로 같은 gradient로 복원된다. 표(3.3)은 이처럼 SPGC 코딩에서 같은 gradient로 decoding 되는 gradient를 구분하여 상위 10개의 gradient를 나타낸 것이다. Start value는 gradient의 시작 컬러 값이며 end value는 gradient가 끝나는 지점의 컬러 값을 나타낸다. 또한 gradient step은 gradient를 구성하는 sub-pixel의 개수이다. 예를 들어 표(3.3)의 rank1에는 표(3.2)의 rank1, rank3, rank5, rank6 외에도 같은 start value, end value, gradient step을 갖는 8개의 gradient를 포함하여 총 12개의 gradient의 중복 횟수를 모두 합한 결과가 9,436 이라는 의미이다. 표(3.3)에서 눈에 띄는 부분은 상위 2개의 gradient의 개수의 합이 총 18,554개로 전체 gradient 중 52%에 해당하는 gradient가 2종류의 gradient로 코딩 된다는 점이다. 또한 상위 10개의 gradient중 6개의 gradient가 gradient의 시작 컬러 값과 끝 컬러 값이 ‘0’ 과 ‘255’ 라는 점이다.

표 3.3 [그림 3.1]영상에서 SPGC 코딩과정에서 같은 gradient로 복원되는 경우를 고려한 상위 10개의 gradient

Rank	Start value	End value	Gradient step	Count
1	0	255	4	9,436
2	0	255	5	9,118
3	0	255	6	1,037
4	0	255	8	895
5	0	255	7	615
6	0	243	5	559
7	0	243	4	549
8	0	215	4	324
9	0	255	9	315
10	0	186	4	283

본 연구에서는 이러한 gradient의 반복성을 고려한 index 코딩 방식을 제안한다. 블록 내에서 빈번하게 나타나는 gradient의 경우 SPGC 코딩 방식을 사용하지 않고 dictionary에 해당 gradient를 저장하여 dictionary의 index로 gradient를 코딩하도록 한다. 이때 dictionary에 있는 모든 gradient를 index로 코딩하는 것이 아니라 가장 빈번하게 나타나는 gradient에 대해서만 index로 코딩하도록 한다. 또한 이러한 과정은 블록 단위로 이루어지기 때문에 블록 내에서 빈번하게 나타나는 gradient의 개수에 따라 index 코딩하는 gradient 개수는 서로 다르며 이러한 gradient의 개수를 결정하는 것이 본 알고리즘의 중요한 요소 중 하나이다.

본 장에서 설명하는 index 코딩 방식은 텍스트 블록 전체에서 나타나는 반복성을 고려한 압축 방식이므로 global index 코딩이라 정의한다.

3.2 Global index 코딩 방법

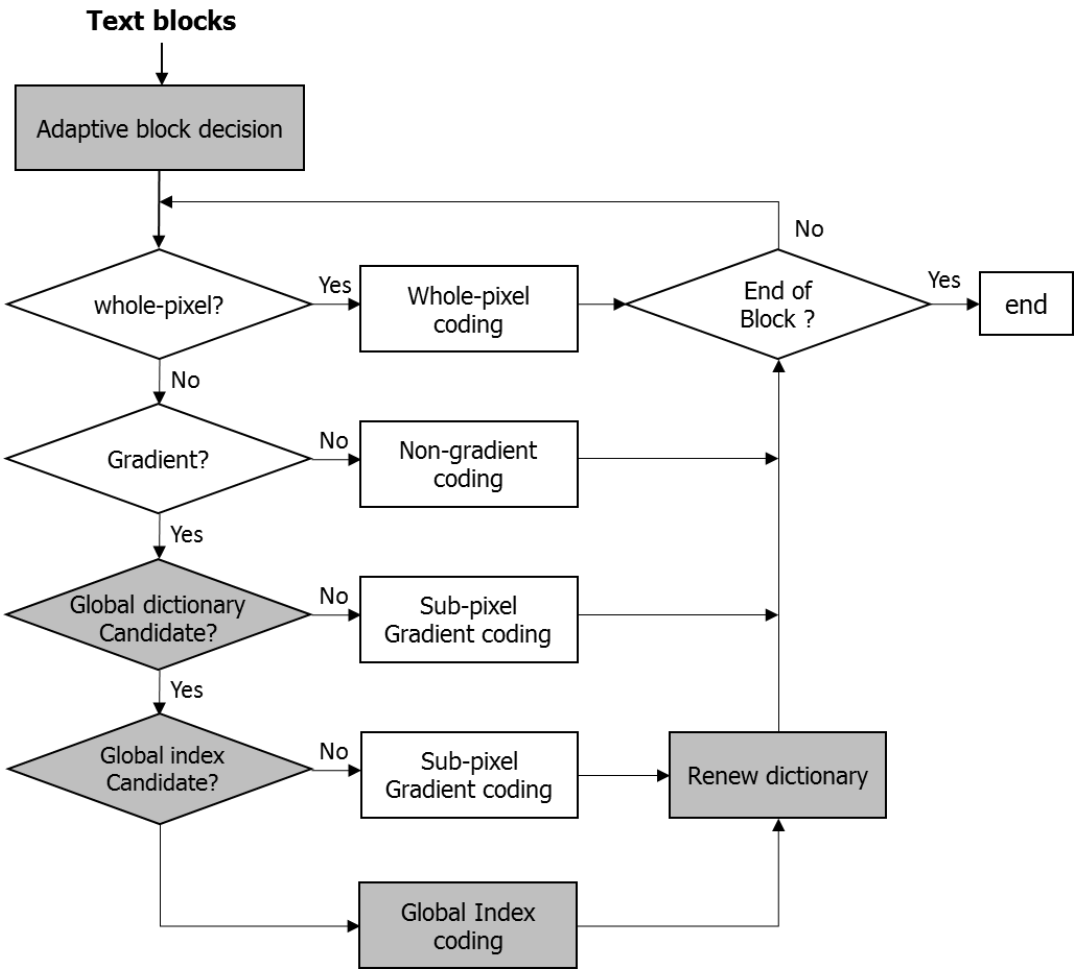


그림 3.2 Global index 코딩 방법

그림(3.2)는 Global index 코딩의 전체적인 흐름도를 나타낸다. 앞에서 설명한 SPGC 코딩의 흐름도에서 추가된 부분을 음영으로 나타내었다.

코딩을 시작하기에 앞서 adaptive block decision 과정을 통해 gradient의 연속성을 최대한 보존하도록 한다. 이후 블록 내에서 gradient가 나타나면, 해당 gradient가 index 코딩 후보인지 확인한다. Index 코딩 후보가 아니라면 기존의 SPGC 코딩 방식을 사용하고 index 코딩 후보라면 현재 gradient가 index 코딩의 조건을 만족하는지 확인한다. 조건을 만족하면 index 코딩을 진행하고 아닐 경우 SPGC 코딩 방식을 수행하도록 한다. Gradient 코딩이 끝나면 index 코딩 후보의 gradient의 경우 dictionary를 갱신하도록 한다.

3.2.1 Adaptive block decision

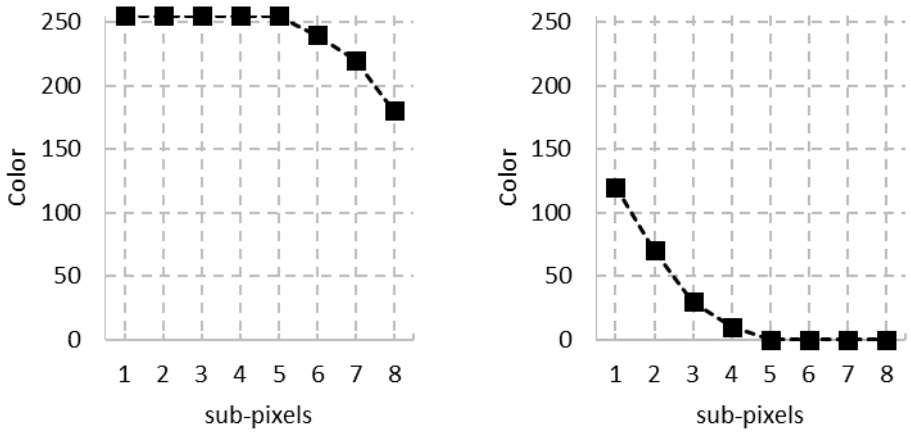


그림 3.3 블록 경계에서의 gradient

본 논문에서 제안하는 알고리즘은 gradient를 직접 압축하기 때문에 gradient의 선형성을 최대한 유지하는 것이 중요하다. 하지만 블록 단위로 압축을 진행하기 때문에 블록을 나누는 과정에서 그림(3.3)과 같이 경계에서 gradient도 함께 2개로 나눠질 수 있다. 이 경우 코딩해야하는 gradient가 1개에서 2개로 늘어나기 때문에 압축 효율에 영향을 미친다. 따라서 코딩을 시작하기에 앞서 텍스트 이웃하는 블록들을 서로 합치는 작업을 수행한다. 블록은 행 단위로 합쳐지며 열 단위로는 합치지 않는다. 그 이유는 그림(3.4)와 같은 compound image에서 텍스트는 다양한 폰트 종류와 크기를 갖기 때문에 블록 단위로 나타나는 gradient의 특성이 서로 다르기 때문이다. 어느 블록에서는 2개의 gradient가 빈번하게 나타날 수 있으며 어느 블록에서는 3개의 gradient가 빈번하게 나타날 수 있다. 이에 따라 각각의 블록에서 index로 코딩할 gradient의 개수 또한 블록마다 다르다. 따라서 같은 행에 위치하고 인접한 텍스트 블록들은 서로 같은 특성을 가질 가능성이 높기 때문에 하나의 블록으로 합쳐서 한번에 코딩을 수행한다. 그림(3.5)는 그림(3.4)의 영상에서 텍스트 블록을 추출한 후 adaptive block decision을 이용해 같은 행의 인접한 블록들을 서로 합친 영상이다. 최종적으로 그림(3.5)의 블록들을 알고리즘의 input으로 이용하여 코딩을 진행한다.

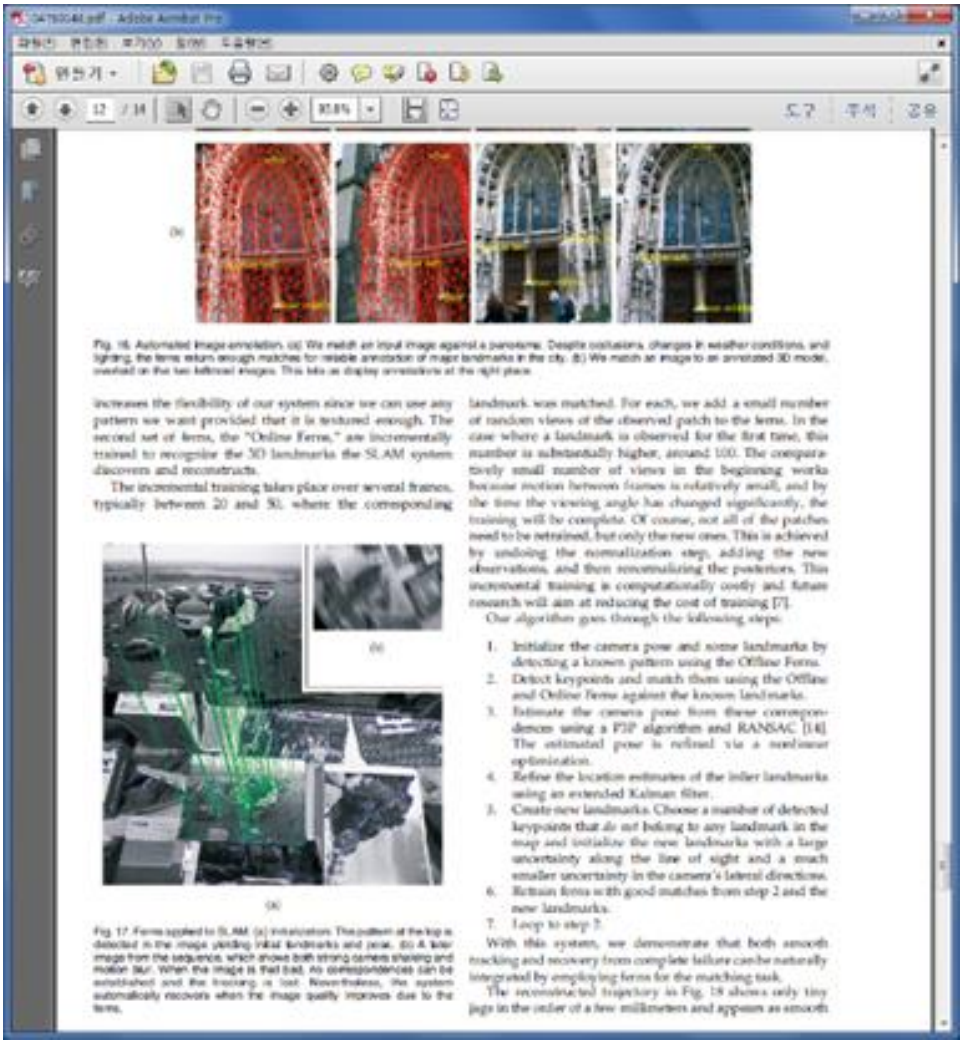


그림 3.4 Compound image

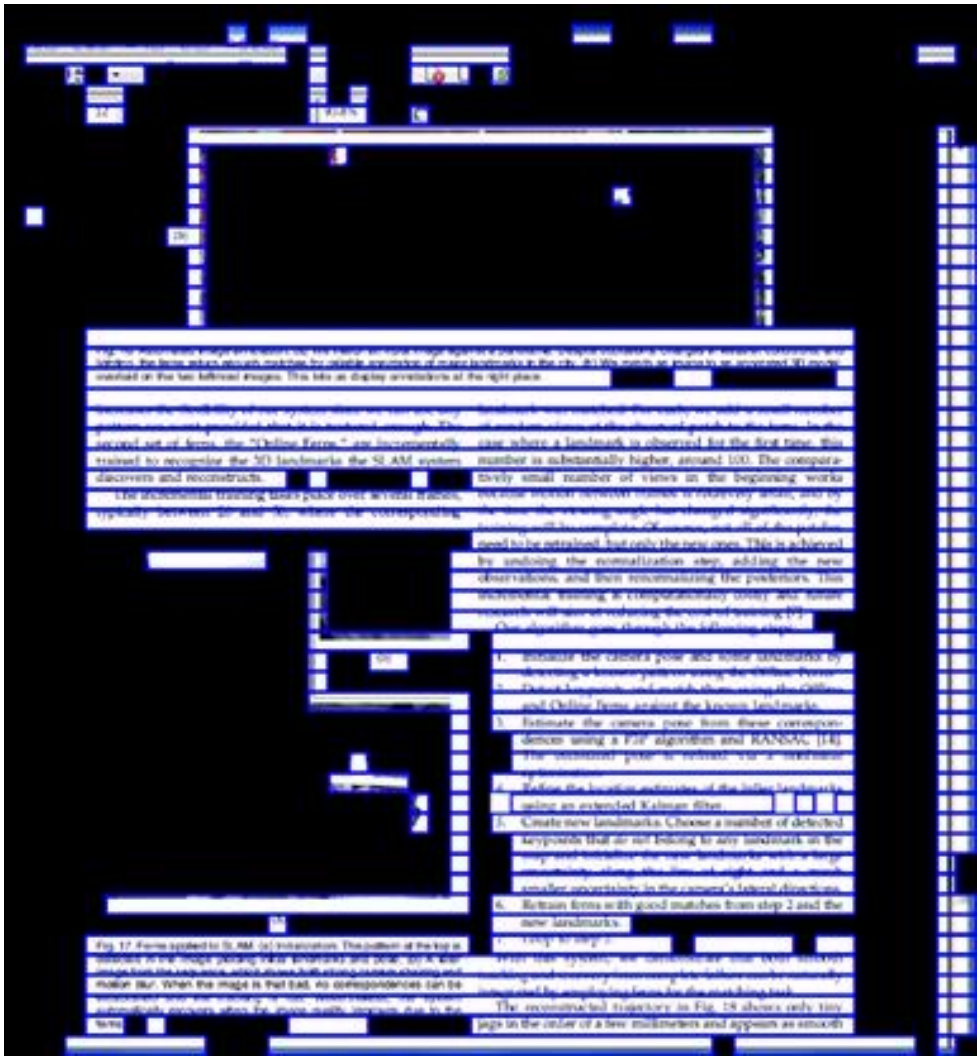


그림 3.5 Compound image의 adaptive block decision 결과

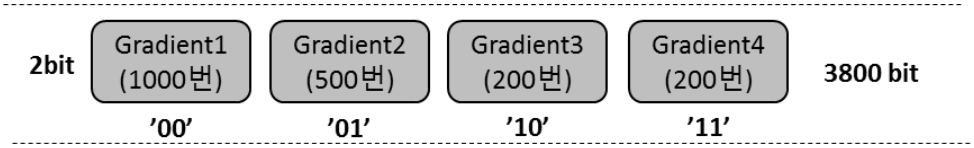
3.2.2 Global index 코딩 후보

텍스트 영상에서 나타나는 모든 gradient를 dictionary에 넣을 경우 dictionary의 크기가 너무 커질 수 있으며, dictionary에 entry가 많으면 dictionary를 탐색하는 시간도 길어진다. 이는 실시간 전송을 필요로

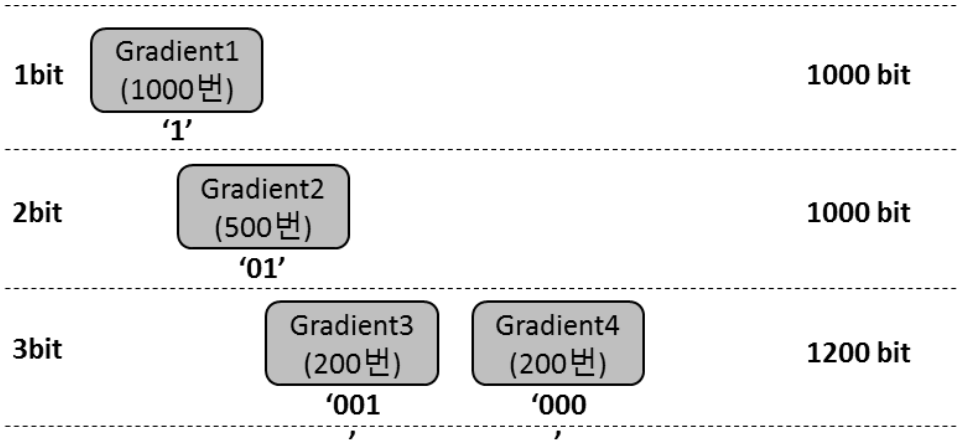
하는 가상화 시스템에 문제를 일으키게 된다. Gradient의 시작점과 끝점의 컬러 값이 각각 ‘0’과 ‘255’인 gradient만을 dictionary에 넣도록 한다. 이는 앞에서 살펴본 것과 같이 텍스트 블록에서 해당 조건을 만족하는 gradient가 가장 많이 반복해서 나타나기 때문이다.

3.2.3 Index 부여 방법 및 index bit 개수 결정 조건

Dictionary에 들어 있는 global dictionary 후보라고 해서 무조건 index로 코딩하지 않는다. 이는 dictionary에 들어있는 gradient 사이에서도 중복 횟수가 서로 확연히 다르기 때문이다. 예를 들어 8개의 gradient를 index로 코딩하기 위해서는 3bit를 필요로 한다. 하지만 이 중 중복 횟수 상위 2개 gradient가 각각 1000번씩 나타나고 나머지 6개의 gradient가 각각 10번씩 나타난다면 상위 2개의 gradient에 대해서 1bit로 각각 ‘0’, ‘1’ 코딩하고 나머지 6개의 gradient는 SPGC 코딩 방식으로 코딩하는 것이 8개 의 모든 gradient를 index 3bit로 코딩하는 것보다 높은 압축 효율을 갖는다. 만약 8개의 gradient 중 상위 4개의 gradient가 각각 1000, 500, 200, 200 번씩 나타나고 하위 4개의 gradient가 10번씩 나타난다면 그림(3.6)의 (a)와 같이 index bit로 2개의 최소 bit를 사용해서 상위 4개의 gradient를 각각 ‘00’, ‘01’, ‘10’, ‘11’로 index 코딩하고 하위 4개의 gradient는 SPGC 방식으로 코딩 할 수 있다. 하지만 이렇게 코딩하는 것보다 (b)와 같이 3개의 bit를 이용하여 상위 4개의 gradient에 대해 각각 ‘1’, ‘01’, ‘001’, ‘000’으로 variable length code 방식으로 index를 부여하고, 나머지 4개의 gradient는 마찬가지로 SPGC 코딩하는 방식이 더 높은 효율을 갖는다.



(a) 최소 bit로 index 부여



(b) 중복 횟수를 고려하여 상향식 variable length code 방식으로 index 결정

그림 3.6 Dictionary의 gradient에 index를 결정하는 방식

실제로 텍스트 블록에서는 gradient의 중복 횟수가 계단식으로 서로 간에 큰 차이를 갖기 때문에 (b)와 같은 트리 형태로 index bit를 부여하는 것이 효율적이다. 물론 경우에 따라서는 (a)와 같이 수평적인 구조를 가져가는 것이 효율적일 수도 있지만, 가상화 시스템의 실시간성을 고려하여 index bit 개수 결정 조건을 간단히 가져가기

위해서 (b)와 같은 상향식 트리 형태의 variable length code로 bit를 부여하도록 한다. 다음은 최적의 bit 개수 결정 조건에 대해 설명하도록 한다.

Index bit 개수에 따라 gradient를 코딩하는데 사용되는 총 bit를 비교하여 최적의 index bit 개수를 선택한다. 표(3.4)는 5개의 gradient를 중복 횟수(count)에 따라 순서대로 나열한 dictionary의 모습이다. Gradient1은 x_1 번의 가장 많은 중복 횟수를 갖으며 gradient5는 x_5번의 가장 적은 중복 횟수를 갖는다.

표 3.4 Dictionary에 저장된 상위 5개 gradient의 중복 횟수

Gradient1	x_1
Gradient2	x_2
Gradient3	x_3
Gradient4	x_4
Gradient5	x_5

표(3.5)는 표(3.4)의 dictionary에 있는 gradient에 1~4bit를 할당하여 코딩할 경우 총 필요한 bit를 나타낸다. 1bit를 할당할 경우 x_1, x_2는 1bit로 index 코딩이 가능하고 나머지 3, 4, 5 gradient는 SPGC 코딩 방식으로 코딩한다. SPGC는 gradient를 구성하는 sub-pixel의 개수를 표현하기 위한 'N_{g_sub}' bit와 gradient의 끝이 0이나 255로 끝난다는 것을 알려주기 위한 1bit 총 '$N_{g\_sub}+1$' bit로 gradient를 코딩할 수 있다. 따라서 3, 4, 5 gradient를 코딩하기 위해서는

$(N_{g\_sub}+1) * (x_3 + x_4 + x_5)$ bit를 필요로 한다. 나머지 행들도 이와 같은 방법으로 index 코딩에 2, 3, 4bit를 할당하여 gradient를 코딩하는 경우 총 필요한 bit를 나타낸다. Index에 1bit를 할당하여 gradient를 코딩하는데 필요한 bit가 index에 2bit를 할당하여 코딩하는 경우보다 많은 bit를 필요로 할 경우 2bit를 할당하여 코딩한다. 만약 2bit보다 3bit를 할당하였을 경우 더 적은 bit를 필요로 하면 다시 3bit로 코딩한다.

표 3.5 Index bit 개수에 따라 gradient 코딩에 필요한 bit

Index 코딩 할당 bit	Gradient1~5를 코딩하기 위해 필요한 bit
1bit	$x_1 + x_2 + (N_{g\_sub}+1) * (x_3+x_4+x_5)$
2bit	$x_1 + 2x_2 + 2x_3 + (N_{g\_sub}+1) * (x_4+x_5)$
3bit	$x_1 + 2x_2 + 3x_3 + 3x_4 + (N_{g\_sub}+1) * (x_5)$
4bit	$x_1 + 2x_2 + 3x_3 + 4x_4 + 4x_5$

표(3.6)은 $N_{g\_sub} = 4$ 일 경우 최적의 index bit 개수 선택 조건을 나타낸다. 각각의 bit 개수 선택 조건은 해당 gradient의 count 값들을 간단하게 비교함으로써 결정할 수 있다.

이와 같은 bit 선택 과정은 encoding 과정에서 dictionary의 count 값의 변화에 따라 실시간으로 선택하게 된다. 따라서 코딩 초기에는 count 값의 비율이 제대로 자리 잡히기 전이므로 위와 같은 조건으로 bit를 선택하면 선택되는 bit가 계속해서 변할 수 있다. 초기에는 무조건 1bit 트리를 이용해서 dictionary의 상위 2개 gradient에 대해서만

index 코딩을 수행하고 나머지는 SPGC 코딩을 수행한다. 이후에 상위 3번째 gradient의 count 값이 20을 넘어가면 2bit 확장 조건을 적용하여 조건을 만족하면 2bit index 코딩으로 상위 3개의 gradient에 대해 index 코딩을 적용한다. 이후에 상위 4번째 gradient도 count 값이 20을 넘어가면 마찬가지로 bit 개수 선택 조건을 적용하여 조건을 만족하면 3bit index 코딩으로 4개의 gradient를 코딩하도록 한다. 이와 같이 20개의 임계 값을 설정함으로써 count 값이 낮을 때 최적의 bit를 선택하는 과정에서 나타나는 bit 변동을 막을 수 있다.

표 3.6 N_{g_sub} = 4일 경우 최적의 index bit 개수 선택 조건

Index 코딩 할당 bit	Bit 개수 선택 조건
1bit	2bit 코딩에 필요한 bit > 1bit 코딩에 필요한 bit $x_2 \geq 3x_3$
2bit	3bit 코딩에 필요한 bit > 4bit 코딩에 필요한 bit $x_3 \geq 2x_4$
3bit	4bit 코딩에 필요한 bit > 5bit 코딩에 필요한 bit $x_4 \geq x_5$
4bit	3bit 선택 조건인 '$x_4 \geq x_5$'는 무조건 만족하므로 4bit 코딩을 선택하는 경우는 없음.

이와 같은 최적의 bit 선택은 decoding 과정에서도 동일하게 재연하여 gradient를 복원한다.

3.2.4 Dictionary 생성 방식

Dictionary에는 index 코딩에 필요한 count 값을 실시간으로 저장하고 encoding 과정에서 gradient를 복원하기 위해 필요한 정보를 저장해야 한다.

SPGC 코딩 방식에서는 gradient를 복원할 때 gradient의 시작과 끝 지점의 컬러 정보와 gradient의 sub-pixel 개수를 이용해서 일정한 기울기로 gradient의 선형적인 특성을 복원한다. 하지만 실제 텍스트 영상에 존재하는 gradient는 일정한 기울기를 갖는 1차원 직선이 아니다. [10]에서는 실제 gradient와 최대한 비슷한 형태로 gradient를 복원하기위해 fitting 기법을 사용하지만 특정 gradient에서는 복원과정에서 fitting 기법을 사용할 경우 오히려 단순히 1차원 직선 형태로 복원하는 경우보다 오차율이 커지는 경우도 발생한다. 따라서 본 논문에서는 1차원 직선 형태로 복원할 경우 발생하는 오차율을 줄이기 위해 gradient를 dictionary에 저장할 때 선형적인 특성을 복원하기 위해 필요한 gradient의 정보(start value, end value, gradient step)를 저장하는 것이 아니라 gradient를 이루는 sub-pixel의 컬러 값들을 저장하도록 한다. 앞의 그림(3.1)의 텍스트 영상에서 에서 '0' 과 '255' 를 시작점과 끝점의 컬러 값으로 갖는 index 후보 gradient 중 gradient를 이루는 sub-pixel의 개수가 5개인 gradient는 그림(3.2)의 rank1, rank3, rank5, rank6 외에도 8개나 존재한다고 설명하였다. 따라서 decoding 과정에서 총 12종류의 gradient를 dictionary에 저장된 하나의 gradient로 복원하기 때문에 dictionary에 gradient를 저장할 때 12종류의 gradient의 sub-pixel 값들을 반영해서 저장하여야 한다. 단순히 12개 gradient의 평균을 계산하여

dictionary에 저장하는 방식은 각각 gradient가 발생하는 횟수에 대한 고려가 없으므로 오차율을 최소화한다고 보기 힘들다. 따라서 오차율을 최소화하기 위해서는 각각 gradient의 발생 횟수에 따른 가중치 평균을 이용하여 저장한다. Dictionary는 encoding 과정에서 실시간으로 생성하기 때문에 가중치 평균 또한 gradient를 코딩하는 순간마다 실시간으로 sub-pixel 컬러 값의 가중치 평균을 계산하여 dictionary에 저장한다. 식(3.1)은 dictionary의 gradient를 업데이트시키는 방법을 나타낸다.

$$Count_{new\_dic} = Count_{dic} + 1$$
$$Color_{new\_dic\_i} = \frac{(Color_{dic\_i} * Count_{dic}) + Color_{now\_i}}{Count_{dic\_i} + 1} \quad (3.1)$$

$Count_{dic}$ 은 현재 dictionary에 저장된 gradient의 count 값이며, $Color_{dic_i}$은 현재 dictionary에 저장된 gradient의 i번째 sub-pixel 컬러 값을 의미한다. $Count_{new_dic}$은 dictionary에 업데이트할 gradient의 count 값이며, $Color_{new_dic_i}$은 dictionary에 업데이트할 gradient의 새로운 i번째 sub-pixel 컬러 값을 의미한다. $Color_{now_i}$은 현재 코딩하려는 gradient의 i번째 sub-pixel 컬러 값이다. Count 값은 gradient가 하나씩 추가될 때 마다 '1' 씩 증가하며 gradient의 sub-pixel 컬러 값은 현재 dictionary의 count값으로 가중치평균을 통해 저장된다.

3.3 Global index 코딩 동작

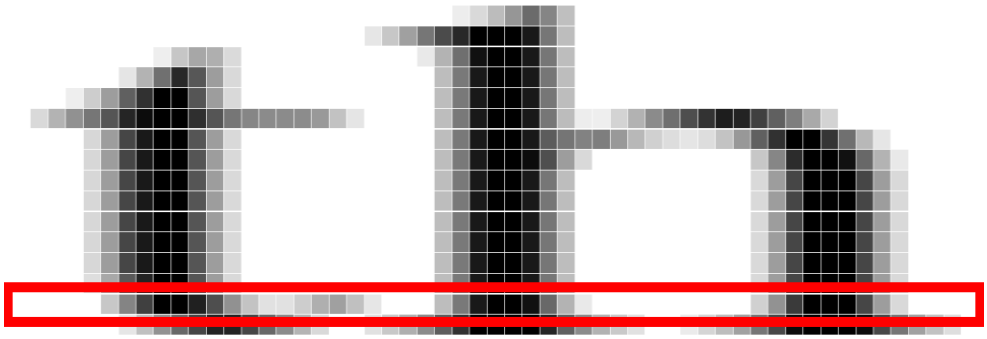


그림 3.7 De-colorization 과정을 거친 text 블록

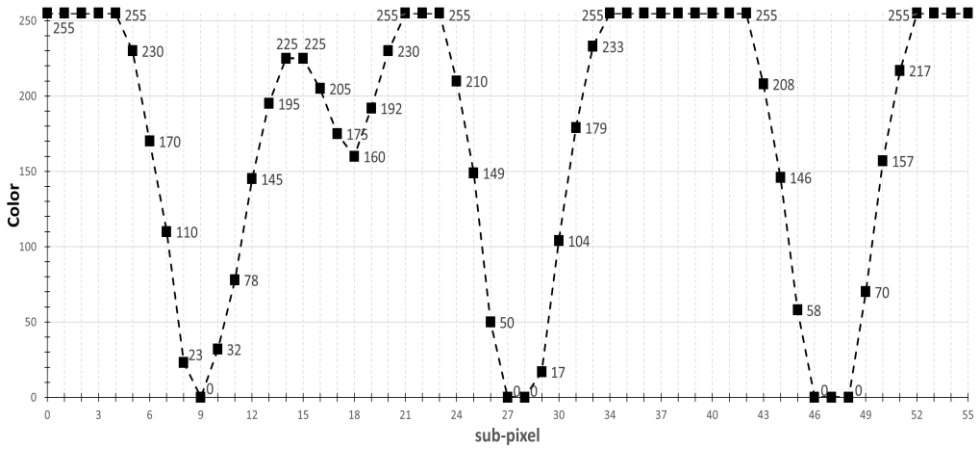


그림 3.8 De-colorization 과정을 거친 text 블록의 sub-pixel 컬러 값

다음은 index 코딩 동작을 단계적으로 설명하고자 한다. 그림(3.2)의 흐름도에서 whole-pixel 코딩과 non-gradient 코딩에 방법에 대해서는 [10]에 제안하는 방식을 그대로 사용하기 때문에 본 절에서는

자세히 설명하지 않고 gradient 코딩 동작에 대해 중점적으로 다루어진다.

그림(3.7)은 de-colorization 과정을 거친 텍스트 블록을 나타내며, 그림(3.8)은 그림(3.7)에서 붉은 박스로 표시된 하나의 라인의 텍스트 컬러 값들을 나타낸 것이다. 텍스트 영역이 de-colorization을 통해 gradient의 경향성을 복원한 것을 알 수 있다. 텍스트가 있는 부분에서 컬러 값이 0으로 감소하는 방향의 gradient를 갖으며, 배경이 있는 부분에서는 컬러 값이 255로 증가하는 방향의 gradient를 갖는다.

표 3.7 Global index 방식에서 gradient가 저장된 dictionary

Rank	Index	Sub-pixel value	Count
1	'0'	20 107 178 234	42
2	'1'	32 58 142 182 223	35
3	SPGC	54 148 209	18
4	SPGC	84 182	13

다음은 이와 같은 하나의 라인에 속한 gradient를 코딩하는 과정에 대해 설명한다. 현재 global index 코딩 과정에서 dictionary는 표(3.7)과 같다고 가정하며, dictionary에는 현재 시작과 끝이 '0'과 '255'인 gradient의 sub-pixel의 개수에 따라 저장되어 있다. 첫 번째 열은 count 값에 따른 gradient의 rank를 나타내며 두 번째 열은 gradient 압축에 사용되는 index를 나타낸다. 현재 rank3 gradient의 count 값이 아직 20을 넘지 않기 때문에 상위 2개(rank1, rank2)의 entry에 대해서만 각각 '0', '1'을 사용하여 index 코딩하고

나머지 하위 2개(rank3, rank4) entry는 SPGC 코딩한다. Index 코딩하는 entry에 대해서는 dictionary에서 음영으로 구분하였다. 세 번째 열은 gradient를 구성하는 sub-pixel의 값들을 나타내며 이 값들은 시작과 끝이 '0'과 '255'인 gradient를 코딩할 때마다 가중치 평균을 통해 업데이트 된다.

3.3.1 Global index 코딩 후보인 gradient의 global index코딩 - (1)

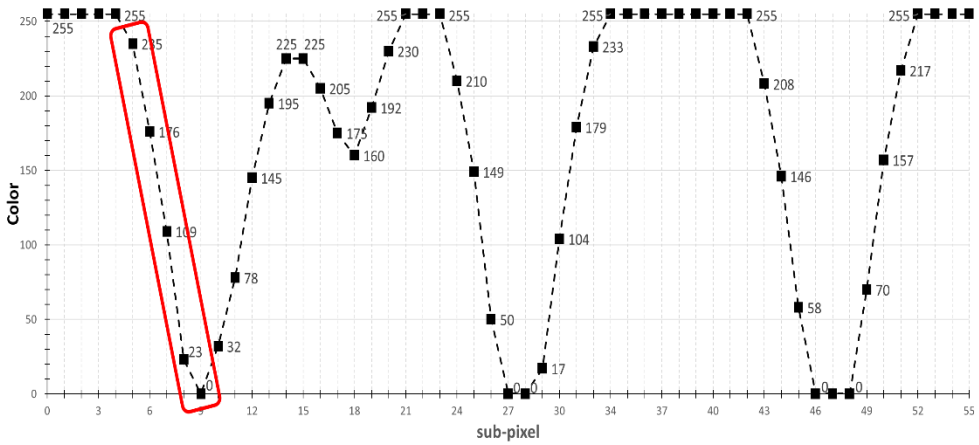


그림 3.9 Global index 코딩 후보인 gradient (1)

표 3.8 (3.3.1)의 gradient 코딩 전 dictionary

Rank	Index	Sub-pixel value	counts
1	'0'	20 107 178 234	42
2	'1'	32 58 142 182 223	35
3	SPGC	54 148 209	18
4	SPGC	84 182	13

그림(3.9)에 표시된 gradient는 시작과 끝점의 컬러 값이 각각

‘255’와 ‘0’ 이기 때문에 index 코딩 후보에 해당한다. 또한 sub-pixel의 개수가 4개이므로 표(3.8)의 현재 dictionary에서 음영으로 표시된 entry중 rank1에 해당한다. 따라서 해당 gradient는 index 코딩을 수행한다. 코딩 되는 bit는 SPGC와 index 코딩 중 index 코딩을 수행한다는 것을 나타내기 위한 1bit(‘1’)와 rank1 의 index 1bit(‘0’)으로 총 2bit (‘10’) 이 코딩 된다.

다음으로 gradient가 index 코딩 후보이므로 dictionary를 갱신하여야 한다. Gradient를 이루는 sub-pixel value [235, 176, 109, 23]이 현재 감소하는 방향이므로 gradient가 증가하는 방향으로 변환하여 [23, 109, 176, 235]를 dictionary에 업데이트 한다. 우선 sub-pixel value를 업데이트 하기 위해서 현재 dictionary rank1의 sub-pixel value [20, 107, 178, 234]와 현재 코딩하는 gradient의 sub-pixel value [23, 109, 176, 235]를 현재 rank1의 count 값으로 가중치를 두어서 42:1의 가중치 평균을 구한다. 가중치 평균 값인 [20.07, 107.05, 177.95, 234.02]를 최종적으로 rank1의 sub-pixel value에 저장하고 count 값을 ‘42’에서 ‘43’으로 ‘1’ 증가 시키면 현재 gradient에 대한 dictionary의 업데이트가 끝난다.

현재 rank3의 count 값이 아직 20을 넘지 않았으므로 bit 결정 과정에 영향을 미치지 않는다. 따라서 여전히 상위 2개(rank1, rank2)의 entry만 index 코딩 entry에 해당한다. 표(3.9)는 현재 업데이트된 dictionary를 나타낸다.

표 3.9 (3.3.1)의 gradient 코딩 후 업데이트된 dictionary

Rank	Index	Sub-pixel value	count
1	'0'	20.07 107.05 177.95 234.02	43
2	'1'	32 58 142 182 223	35
3	SPGC	54 148 209	18
4	SPGC	84 182	13

3.3.2 Global index 코딩 후보가 아닌 gradient의 SPGC 코딩

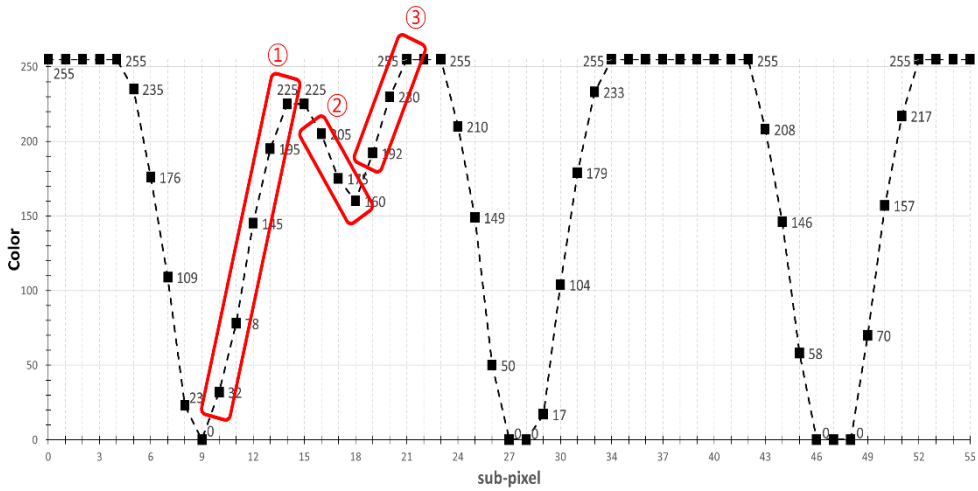


그림 3.10 Global index 코딩 후보가 아닌 gradient

그림(3.10)에 표시된 ①번 gradient는 시작 컬러 값은 '0' 이지만 끝 컬러 값이 '255'가 아니므로 index 코딩 후보가 아니다. 따라서 SPGC 코딩을 진행한다. Gradient의 끝점이 '0' 이나 '255'로 끝나지 않기 때문에 끝점의 컬러 값을 비트스트림에 넣어줘야 한다.

코딩 되는 bit는 SPGC 코딩을 나타내는 index ‘1(1bit)’, gradient를 구성하는 sub-pixel의 개수 ‘5’를 나타내는 ‘0101(4bit)’, gradient의 끝점 컬러 값이 ‘255’ 또는 ‘0’으로 끝나지 않으므로 ‘1(1bit)’와 끝점 컬러 값 225 ‘11100001(8bit)’으로 총 14bit ‘1 0101 1 11100001’이 코딩 된다.

Gradient가 index 코딩 후보가 아니므로 dictionary는 갱신하지 않아도 된다. 따라서 dictionary는 표(3.10)과 같이 이전과 같은 상태로 유지된다.

표 3.10 (3.3.2)의 gradient 코딩 후 업데이트된 dictionary

Rank	Index	Sub-pixel value	count
1	‘0’	20.07 107.05 177.95 234.02	43
2	‘1’	32 58 142 182 223	35
3	SPGC	54 148 209	18
4	SPGC	84 182	13

②번 gradient도 마찬가지로 끝점의 컬러 값이 ‘0’이나 ‘255’가 아니므로 index 코딩 후보가 아니다. 따라서 ①번 gradient와 마찬가지로 sub-pixel gradient 코딩 방식을 사용한다. ①번 gradient 코딩 결과에서 gradient를 구성하는 sub-pixel의 개수 ‘3(0011)’, 끝점의 컬러 값 ‘150(10010110)을 반영하여 총 14bit ‘1 0011 1 10010110’이 코딩 된다.

③번 gradient의 경우 끝점은 ‘255’이지만 시작점의 컬러 값이 ‘0’이 아니므로 index 코딩 후보가 아니다. 따라서 ①, ② gradient와

마찬가지로 sub-pixel gradient 코딩 방식을 사용한다. 하지만 끝점이 '255' 이므로 끝점의 컬러 값을 따로 코딩할 필요가 없다. Gradient를 구성하는 sub-pixel의 개수 '3(0011)'과 끝점 컬러 값이 '255' 또는 '0'으로 끝난다는 것을 나타내는 color bit '0(1bit)'을 반영하여 '1 0011 0' 총 6bit가 코딩 된다.

②, ③번 gradient 모두 index 코딩 후보가 아니므로 dictionary는 표(3.9)에서 변하지 않는다.

3.3.3 Global index 코딩 후보인 gradient의 SPGC 코딩 - (1)

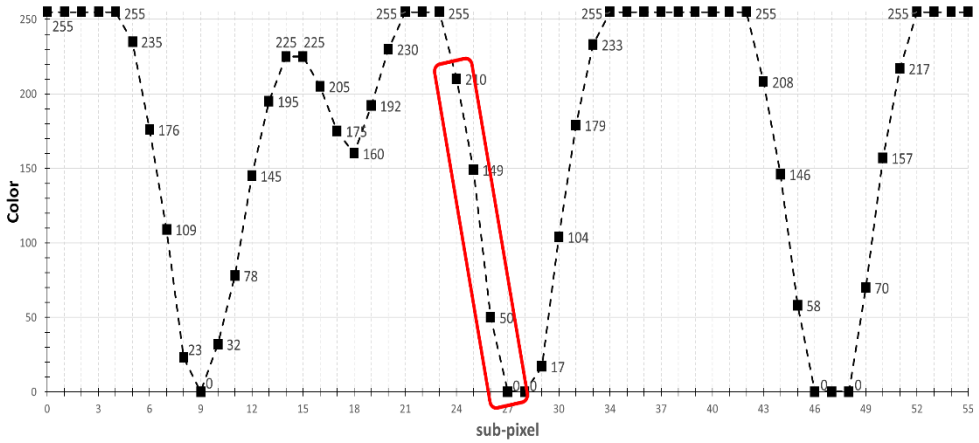


그림 3.11 Global index 코딩 후보인 gradient (2)

그림(3.11)에 표시된 gradient는 시작과 끝점의 컬러 값이 각각 '255'와 '0' 이기 때문에 index 코딩 후보에 해당한다. 하지만 sub-pixel의 개수가 3개이므로 표(3.11)의 현재 dictionary에서 index 코딩 entry(rank1, rank2)에 해당하지 않는다. 따라서 해당 gradient는 SPGC 코딩을 수행한다.

코딩 되는 bit는 SPGC 와 index 코딩 중 SPGC 코딩을 수행한다는 것을 나타내기 위한 1bit(‘0’), gradient를 구성하는 sub-pixel의 개수 ‘3’ 을 나타내는 4bit(‘0011’), gradient의 끝점 컬러 값이 ‘255’ 또는 ‘0’ 으로 끝난다는 것을 나타내는 1bit(‘0’)으로 총 6bit(‘0 0011 0’) 이 코딩 된다.

표 3.11 (3.3.3)의 gradient 코딩 전 dictionary

Rank	Index	Sub-pixel value	count
1	‘0’	20.07 107.05 177.95 234.02	43
2	‘1’	32 58 142 182 223	35
3	SPGC	54 148 209	18
4	SPGC	84 182	13

다음으로 gradient가 index 코딩 후보이므로 index 코딩 entry 여부에 상관없이 dictionary를 갱신하여야 한다. Dictionary에서 sub-pixel의 개수가 ‘3’ 인 entry는 rank3에 위치하므로 rank3을 갱신하도록 한다. 현재 코딩하는 gradient를 이루는 sub-pixel value [50, 149, 210]를 dictionary에 업데이트 한다. 현재 dictionary rank3의 sub-pixel value [54, 148, 209]와 현재 코딩하는 gradient의 sub-pixel value [50, 149, 210]를 현재 rank3의 count 값으로 가중치를 두어서 18:1의 가중치 평균을 구한다. 가중치 평균 값인 [53.79, 148.05, 209.05]을 최종적으로 rank3의 sub-pixel value에 저장하고 count 값을 ‘18’ 에서 ‘19’ 로 ‘1’ 증가시킨다.

현재 rank3의 count 값이 아직 20을 넘지 않았으므로 bit 결정

과정에 영향을 미치지 않는다. 따라서 여전히 상위 2개(rank1, rank2)의 entry만 index 코딩 entry에 해당한다. 표(3.12)는 현재 업데이트된 dictionary를 나타낸다.

표 3.12 (3.3.3)의 gradient 코딩 후 업데이트된 dictionary

Rank	Index	Sub-pixel value	count
1	'0'	20.07 107.05 177.95 234.02	43
2	'1'	32 58 142 182 223	35
3	SPGC	53.79 148.05 209.05	19
4	SPGC	84 182	13

3.3.4 Global index 코딩 후보인 gradient의 Global index 코딩 - (2)

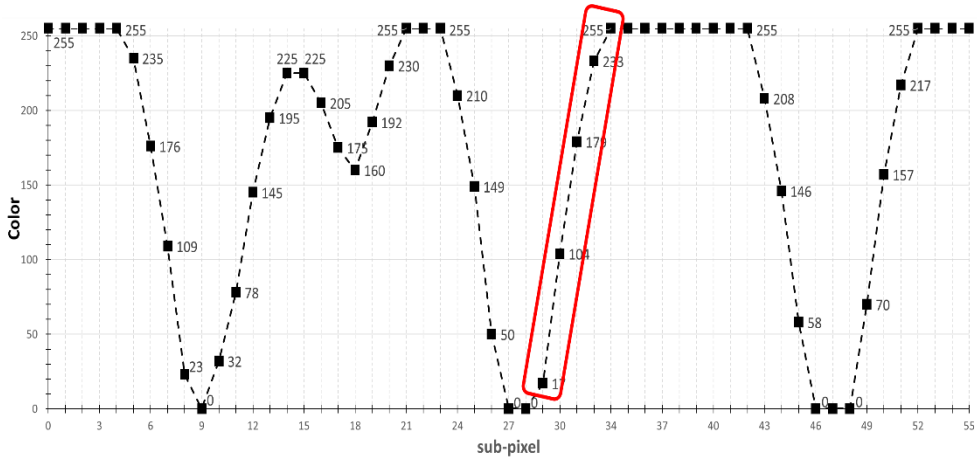


그림 3.12 Global index 코딩 후보인 gradient (3)

그림(3.12)에 표시된 gradient는 시작과 끝점이 각각 '0'과 '255'인 index 코딩 후보 이며 sub-pixel의 개수가 4개이므로 표(3.13)는 현재 dictionary에서 index 코딩 entry인 rank1에 해당한다.

코딩 되는 bit는 SPGC 와 index 코딩 중 index 코딩을 수행한다는 것을 나타내기 위한 1bit('1')와 rank1 의 index 1bit('0')으로 총 2bit ('10') 이 코딩 된다.

표 3.13 (3.3.4)의 gradient 코딩 전 dictionary

Rank	Index	Sub-pixel value	count
1	'0'	20.07 107.05 177.95 234.02	43
2	'1'	32 58 142 182 223	35
3	SPGC	53.79 148.05 209.05	19
4	SPGC	84 182	13

다음으로 gradient가 index 코딩 후보이므로 dictionary를 갱신한다. Gradient를 이루는 sub-pixel value [17, 104, 179, 233]을 dictionary에 업데이트 한다. 현재 dictionary rank1의 sub-pixel value [20.07, 107.05, 177.95, 234.02]와 현재 코딩하는 gradient의 sub-pixel value [17, 104, 179, 233]를 현재 rank1의 count 값으로 가중치를 두어서 43:1의 가중치 평균을 구한다. 가중치 평균 값인 [20, 106.98, 177.96, 234]를 최종적으로 rank1의 sub-pixel value에 저장하고 count 값을 '43'에서 '44'로 '1' 증가 시키면 현재 gradient에 대한 dictionary의 업데이트가 끝난다.

현재 rank3의 count 값이 아직 20을 넘지 않았으므로 bit 결정 과정에 영향을 미치지 않는다. 따라서 여전히 상위 2개(rank1, rank2)의 entry만 index 코딩 entry에 해당한다. 표(3.14)은 현재 업데이트된 dictionary를 나타낸다.

표 3.14 (3.3.4)의 gradient 코딩 후 업데이트된 dictionary

Rank	Index	Sub-pixel value	count
1	'0'	20 106.98 177.97 234	44
2	'1'	32 58 142 182 223	35
3	SPGC	53.79 148.05 209.05	19
4	SPGC	84 182	13

3.3.5 Global index 코딩 후보인 gradient의 SPGC 코딩 - (2)

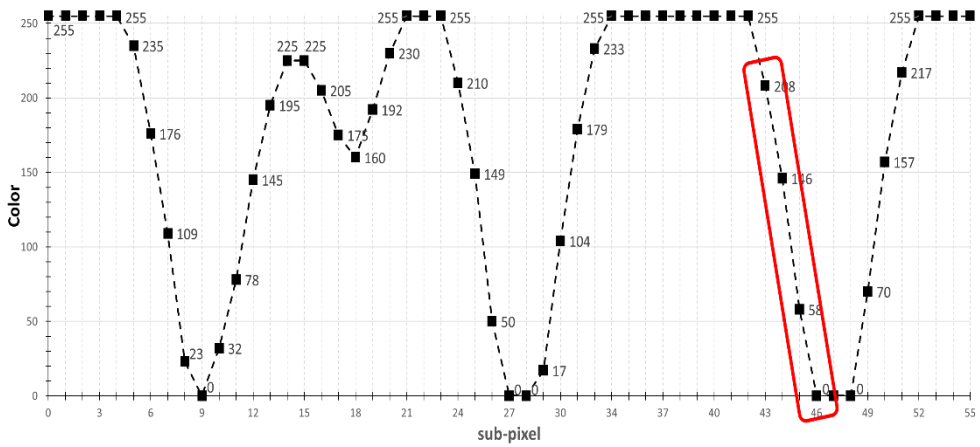


그림 3.13 Global index 코딩 후보인 gradient (4)

그림(3.13)에 표시된 gradient는 시작과 끝점이 각각 ‘255’와 ‘0’인 index 코딩 후보 이다. 하지만 sub-pixel의 개수가 3개이므로 표(3.15)의 현재 dictionary에서 index 코딩 entry가 아니다. 따라서 SPGC 코딩을 수행한다.

코딩 되는 bit는 SPGC 와 index 코딩 중 SPGC 코딩을 수행한다는 것을 나타내기 위한 1bit(‘0’), gradient를 구성하는 sub-pixel의 개수 ‘3’을 나타내는 4bit(‘0011’), gradient의 끝점 컬러 값이 ‘255’ 또는 ‘0’으로 끝난다는 것을 나타내는 1bit(‘0’)으로 총 6bit(‘0 0011 0’) 이 코딩 된다.

표 3.15 (3.3.5)의 gradient 코딩 전 dictionary

Rank	Index	Sub-pixel value	count
1	‘0’	20 106.98 177.97 234	44
2	‘1’	32 58 142 182 223	35
3	SPGC	53.79 148.05 209.05	19
4	SPGC	84 182	13

다음으로 gradient가 index 코딩 후보이므로 index 코딩 entry 여부에 상관없이 dictionary를 갱신하여야 한다. Dictionary에서 sub-pixel의 개수가 ‘3’인 entry는 rank3에 위치하므로 rank3을 갱신하도록 한다. 현재 코딩하는 gradient를 이루는 sub-pixel value [208, 146, 58]의 방향을 증가하는 방향으로 변경하여 dictionary에 업데이트 한다. 현재 dictionary rank3의 sub-pixel value [53.79 148.05 209.05]와 현재 코딩하는 gradient의 sub-pixel value [58,

146, 208]을 현재 rank3의 count 값으로 가중치를 두어서 19:1의 가중치 평균을 구한다. 가중치 평균 값인 [54, 147.95, 209]를 최종적으로 rank3의 sub-pixel value에 저장하고 count 값을 '19'에서 '20'로 '1' 증가시킨다.

현재 Rank3의 count 값이 '20'을 넘었으므로 앞에서 설명한 global index 코딩 bit 결정 조건을 통해 index bit를 결정하여야 한다. 우선 1bit 결정 조건을 통해 계속해서 index로 1bit만 사용할지 확인한다. 현재 rank2의 count 값 '35'가 rank3의 count 값에 '3'을 곱한 '60'보다 작으므로 더 이상 1bit index를 사용할 수 없으며 2bit를 index bit로 사용하여야 한다. 만약 rank4의 count 값도 '20'을 넘었다면 2bit 결정 조건을 통해 2bit를 사용할지 3bit를 사용할지 확인하여야 하지만 현재는 rank4의 count 값이 '13'으로 '20'을 넘지 못했기 때문에 index bit는 2bit를 사용하도록 한다. 따라서 상위 3개(rank1, rank2, rank3)의 entry가 index 코딩 entry에 해당하며 상향식 variable length code 방식으로 각각 '0', '10', '11'을 부여한다. 표(3.16)는 최종적으로 업데이트된 dictionary를 나타낸다. Index 코딩을 수행하는 상위 3개의 entry를 음영으로 표시하였다.

표 3.16 (3.3.5)의 gradient 코딩 후 업데이트된 dictionary

Rank	Index	Sub-pixel value	count
1	'0'	20 106.98 177.97 234	44
2	'10'	32 58 142 182 223	35
3	'11'	54 147.95 209	20
4	SPGC	84 182	13

3.3.6 Global index 후보인 gradient의 global index 코딩 - (3)

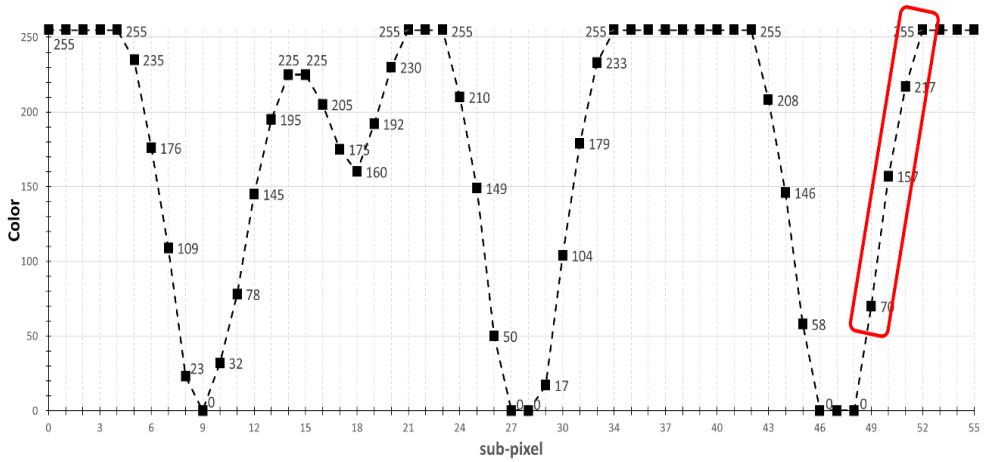


그림 3.14 Global index 코딩 후보인 gradient (5)

표 3.17 (3.3.6)의 gradient 코딩 전 dictionary

Rank	Index	Sub-pixel value	count
1	'0'	20 106.98 177.97 234	44
2	'10'	32 58 142 182 223	35
3	'11'	54 147.95 209	20
4	SPGC	84 182	13

그림(3.14)에 표시된 gradient는 시작과 끝점이 각각 '0'과 '255'인 index 코딩 후보이며 앞서 코딩한 gradient와 마찬가지로 sub-pixel의 개수가 3개이다. 표(3.17)의 현재 dictionary에서 sub-pixel의 개수가 '3'인 entry(rank3)도 index 코딩이 가능하므로 index 코딩을 수행한다.

코딩 되는 bit는 SPGC와 index 코딩 중 index 코딩을 수행한다는 것을 나타내기 위한 1bit(‘1’)와 rank3 의 index 2bit(‘11’)으로 총 3bit (‘111’) 이 코딩 된다.

다음으로 gradient가 index 코딩 후보이므로 dictionary를 갱신한다. Gradient를 이루는 sub-pixel value [70, 157, 217]을 dictionary에 업데이트 한다. 현재 dictionary rank3의 sub-pixel value [54, 147.95, 209]와 현재 코딩하는 gradient의 sub-pixel value [70, 157, 217]를 현재 rank3의 count 값으로 가중치를 두어서 20:1의 가중치 평균을 구한다. 가중치 평균 값인 [54.76, 148.38, 209.38]를 최종적으로 rank3의 sub-pixel value에 저장하고 count 값을 ‘20’에서 ‘21’로 ‘1’ 증가시키면 현재 gradient에 대한 dictionary의 업데이트가 끝난다. 표(3.18)은 최종적으로 업데이트된 dictionary를 나타낸다.

표 3.18 (3.3.6)의 gradient 코딩 후 업데이트된 dictionary

Rank	Index	Sub-pixel value	count
1	‘0’	20 106.98 177.97 234	44
2	‘10’	32 58 142 182 223	35
3	‘11’	54.76 148.38 209.38	21
4	SPGC	84 182	13

본 과정과 같은 방법으로 global index 코딩을 진행한다. 앞의 과정에서는 나타나지 않았지만 dictionary 내에서 entry의 rank는 실시간으로 변경된다. 만약 dictionary를 업데이트하는 과정에서

rank3의 count 값이 rank2의 count 값보다 커지면 서로 위치를 바꿔 저장한다.

모든 텍스트 블록의 압축이 끝나면 최종적으로 생성된 dictionary의 sub-pixel 비트스트림에 저장하고, decoding 과정에서 gradient를 복원하는 과정에 사용한다.

Decoding 과정에서도 encoding 과정과 마찬가지로 dictionary의 count 값에 따라 entry의 위치를 업데이트 하면서 encoding 과정에서 수행했던 dictionary의 rank 및 index 코딩 bit 결정을 재연한다.

3.4 Global index 코딩 실험 결과

Global index를 이용한 코딩은 앞에서 설명한 바와 같이 진행된다. 본 실험 결과는 이러한 global index 코딩 방법의 압축효율성과 화질 저하에 대해서 확인한다. 실험은 본 장에서 제안하는 global index 압축 방법과 비교를 통해 진행된다. 실험은 본 연구에서 제안하는 알고리즘이 compound 영상에서 텍스트 블록을 압축하는 알고리즘이기 때문에 Kim [10]에서 제안하는 SPGC 압축 알고리즘과의 비교를 통해 진행된다. 표준 알고리즘인 JPEG나 H.264 와의 비교는 이미 Kim [10]에서 SPGC 알고리즘이 표준 알고리즘들에 비해 텍스트 블록 압축에서 높은 압축률과 화질을 갖는다는 것을 입증하였기 때문에 본 실험에서는 SPGC 알고리즘과의 비교만을 진행하도록 한다. 실험 영상은 RGB 포맷의 텍스트 영상을 캡처하여 사용하였으며, 실험 결과 또한 RGB 포맷으로 작성되었다. 실험은 simple rendered 텍스트 이미지와 complex rendered 텍스트 이미지 총 2가지 종류의 이미지를 기준으로

진행하였다. Simple rendered 텍스트 이미지의 경우는 텍스트 블록에서 나타나는 sub-pixel의 컬러 값이 10종류를 넘지 않는 이미지로 대표적으로 word 문서프로그램과 webpage 등에서 나타나는 이미지이다. 컬러 값의 종류가 제한되기 때문에 블록 내에서 생성되는 gradient도 2~3개의 대표 gradient가 전체 gradient의 90% 이상을 차지한다. Complex rendered 텍스트 이미지의 경우는 sub-pixel의 컬러 값이 다양한 이미지로 대표적으로 PDF 파일 문서에서 나타나는 이미지이다. 컬러 값이 다양하기 때문에 2절에서 확인한 것처럼 gradient 또한 수백 종류의 다양한 gradient가 텍스트 블록 전 영역에 걸쳐 나타난다.

해상도는 640 × 480 와 1024 × 768 크기의 텍스트 이미지를 사용하였다. Complex rendered 텍스트 이미지의 경우 폰트 크기에 따른 압축률을 확인하기 위해서 2종류의 이미지를 가지고 실험하였다.

그림(3.15)의 (a)는 실험에 사용된 simple rendered 텍스트 이미지이고, (b), (c)는 실험에 사용된 2종류의 complex rendered 텍스트 이미지이다.

Related: Earth's 'twin' will be found in 2013, so start trashing this planet

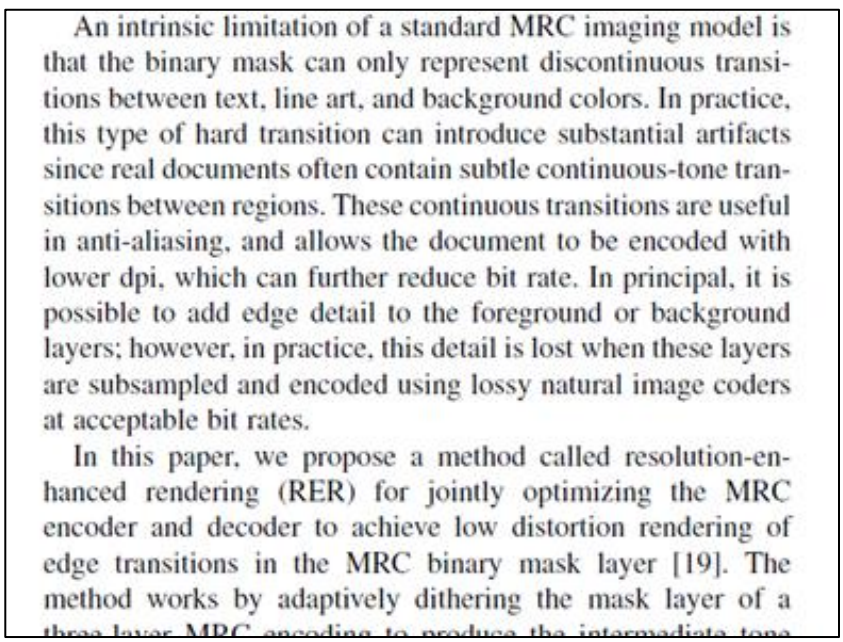
Besides its color, the planet is unusual because of its distance from its sun, GJ 504 or 59 Virginis — nearly nine times the distance Jupiter orbits the sun, or 43.5 astronomical units. (One astronomical unit is the distance from the Earth to the sun, or about 92,955,807 miles) The sheer distance makes astronomers start to question some of the theories they have around how planets form.

"This is among the hardest planets to explain in a traditional planet-formation framework," said Markus Janson, a Hubble postdoctoral fellow at Princeton and one of the researchers on the project. "Its discovery implies that we need to seriously consider alternative formation theories." At the very least, assessing basic assumptions in the current popular planet formation theory would be important, said Janson in a NASA release.

Related: Three new habitable planets discovered — and they're not very far away

Infrared photos of the planet were taken by the Subaru Telescope on Mauna Kea in Hawaii. The approximately 160 million-year-old star system in which the exoplanet resides orbits around the star GJ 504, and is approximately 57 light-years from Earth. Especially exciting is the fact that GJ 504 is about only 1/30th the age of our own sun, says Michael McElwain, a member of the discovery team at NASA's Goddard Space Flight Center. "Studying these systems is a little like seeing our own planetary system in its youth."

(a) Simple rendered 텍스트 (Webpage, 640×480)



An intrinsic limitation of a standard MRC imaging model is that the binary mask can only represent discontinuous transitions between text, line art, and background colors. In practice, this type of hard transition can introduce substantial artifacts since real documents often contain subtle continuous-tone transitions between regions. These continuous transitions are useful in anti-aliasing, and allows the document to be encoded with lower dpi, which can further reduce bit rate. In principal, it is possible to add edge detail to the foreground or background layers; however, in practice, this detail is lost when these layers are subsampled and encoded using lossy natural image coders at acceptable bit rates.

In this paper, we propose a method called resolution-enhanced rendering (RER) for jointly optimizing the MRC encoder and decoder to achieve low distortion rendering of edge transitions in the MRC binary mask layer [19]. The method works by adaptively dithering the mask layer of a three layer MRC encoding to produce the intermediate tone

(b) Complex rendered 텍스트 (1), (PDF, 640×480)

Fig. 2. Derivative based filters. (a) Gaussian derivatives up to fourth order. (b) Complex filters up to sixth order. Note that the displayed filters are not weighted by a Gaussian, for figure clarity.

filters and wavelets [27] are frequently explored in the context of texture classification.

2.3 Differential Descriptors

A set of image derivatives computed up to a given order approximates a point neighborhood. The properties of local derivatives (*local jet*) were investigated by Koenderink and van Doorn [20]. Florack et al. [11] derived differential invariants, which combine components of the *local jet* to obtain rotation invariance. Freeman and Adelson [12] developed steerable filters, which steer derivatives in a particular direction given the components of the *local jet*. Steering derivatives in the direction of the gradient makes them invariant to rotation. A stable estimation of the derivatives is obtained by convolution with Gaussian derivatives. Fig. 2a shows Gaussian derivatives up to order 4.

Baumberg [2] and Schaffalitzky and Zisserman [37] proposed using complex filters derived from the family $K(x, y, \theta) = f(x, y)\exp(i\theta)$, where θ is the orientation. For the function $f(x, y)$, Baumberg uses Gaussian derivatives and Schaffalitzky and Zisserman apply a polynomial (cf., Section 3.2 and Fig. 2b). These filters differ from the Gaussian derivatives by a linear coordinates change in filter response domain.

2.4 Other Techniques

Generalized moment invariants have been introduced by Van Gool et al. [43] to describe the multispectral nature of the image data. The invariants combine central moments defined by $M^a_{pq} = \int\int_\Omega x^p y^q [I(x, y)]^a dxdy$ of order $p + q$ and degree a. The moments characterize shape and intensity distribution in a region Ω. They are independent and can be easily computed for any order and degree. However, the moments of high order and degree are sensitive to small geometric and photometric distortions. Computing the invariants reduces the number of dimensions. These descriptors are therefore more suitable for color images where the invariants can be computed for each color channel and between the channels.

3 Experimental Setup

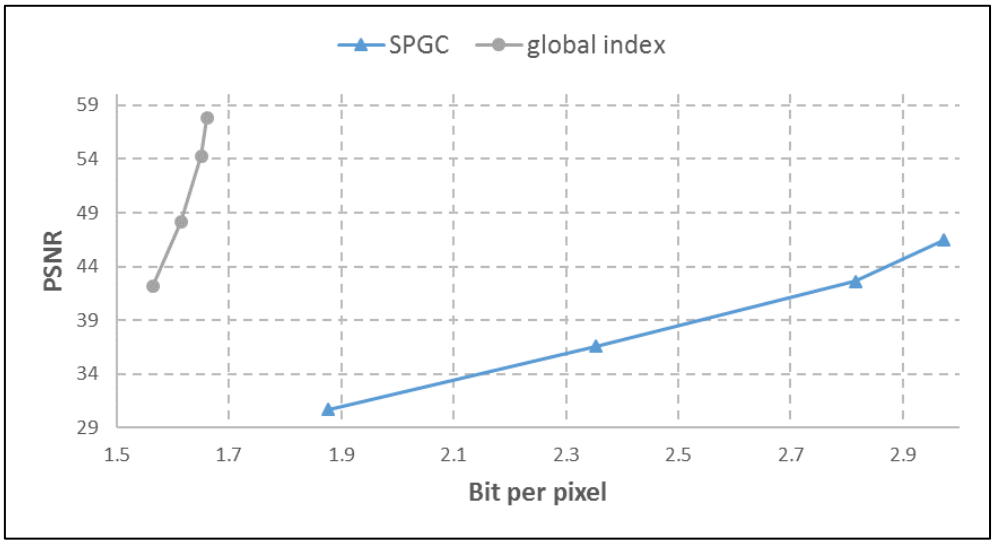
In the following, we first describe the region detectors used in our comparison and the region normalization necessary for computing the descriptors. We then give implementation details for the evaluated descriptors. Finally, we discuss the evaluation criterion and the image data used in the tests.

3.1 Support Regions

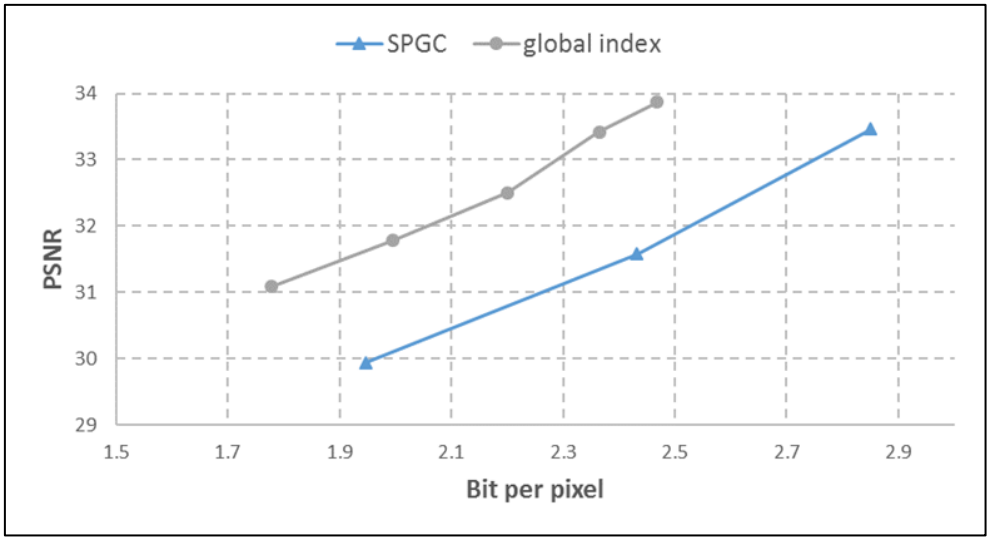
Many scale and affine invariant region detectors have been recently proposed. Lindeberg [23] has developed a scale-invariant "blob" detector, where a "blob" is defined by a maximum of the normalized Laplacian in scale-space. Lowe [25] approximates the Laplacian with difference-of-Gaussian (DoG) filters and also detects local extrema in scale-space. Lindeberg and Gårding [24] make the blob detector affine-invariant using an *affine adaptation* process based on the second moment matrix. Mikolajczyk and Schmid [29], [30] use a multiscale version of the Harris interest point detector to localize interest points in space and then employ Lindeberg's scheme for scale selection and affine adaptation. A similar idea was explored by Baumberg [2] as well as Schaffalitzky and Zisserman [37]. Tuytelaars and Van Gool [42] construct two types of affine-invariant regions, one based on a combination of interest points and edges and the other one based on image intensities. Matas et al. [28] introduced Maximally Stable Extremal Regions extracted with a watershed like segmentation algorithm.

(C) complex rendered 텍스트 (2), (PDF, 800×720)

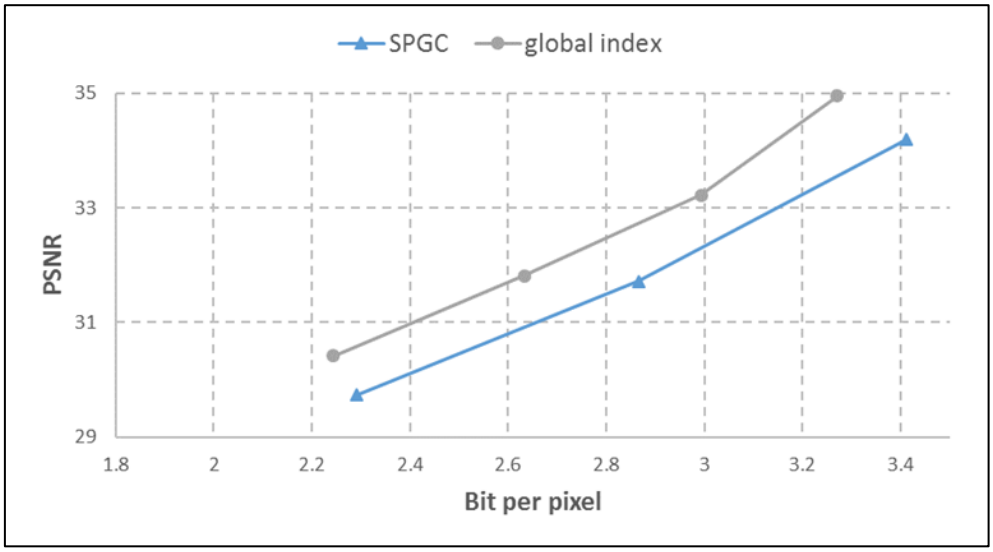
그림 3.15 성능 비교를 위한 실험 영상



(a) Simple rendered 텍스트



(b) Complex rendered 텍스트 (1)



(C) Complex rendered 텍스트 (2)

그림 3.16 성능 비교 실험 결과

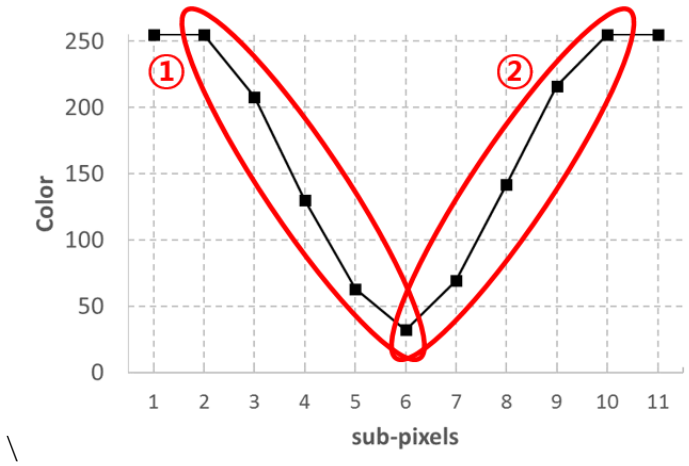
Global index 코딩 방식과 SPGC 코딩 방식을 적용하여 최종 압축 효율과 영상의 화질에 대한 비교를 그림(3.15)에 나타내었다.

그림(3.16)의 (a)와 같은 simple rendered 텍스트 이미지의 경우 기존의 SPGC 방식과 비교해서 동일한 화질에서 약 80%의 압축 효율 개선에 대한 결과를 확인할 수 있었다. Simple rendered 텍스트 이미지의 경우 2~3개의 대표 gradient가 전체 텍스트 블록에서 90% 이상 나타나기 때문에 global index 방식으로 압축 효율을 크게 개선할 수 있었다. 또한 sub-pixel의 컬러 값까지 완전히 같은 gradient가 반복적으로 나타나기 때문에 dictionary에 가중치평균을 통해 저장한 gradient의 sub-pixel 컬러 값이 실제 gradient와 큰 차이가 없으므로 화질 또한 SPGC와 비교해서 높은 성능 향상을 보여주고 있다.

그림(3.16)의 (b), (c)와 같은 complex rendered 텍스트 이미지의 경우에도 SPGC에 비하여 1~2 dB 의 화질 개선 효과를 확인할 수 있었다. 또한 폰트의 크기가 큰 (b)의 경우가 (c)보다 조금 더 높은 성능향상을 보였다. 이는 폰트의 크기가 클수록 gradient의 시작 컬러 값과 끝 컬러 값이 '0' 과 '255' 인 global index 코딩 후보가 많이 나타나고, 폰트의 크기가 작을수록 텍스트를 구성하는 pixel의 개수가 적어 gradient의 컬러 값이 '255' 에서 '0' 까지 내려가지 못하고 다시 '255' 로 올라가는 경우가 나타난다. 따라서 폰트의 크기가 클수록 global index 코딩 후보가 많이 나타나기에 좀 더 높은 성능의 개선 효과가 나타남을 확인할 수 있었다.

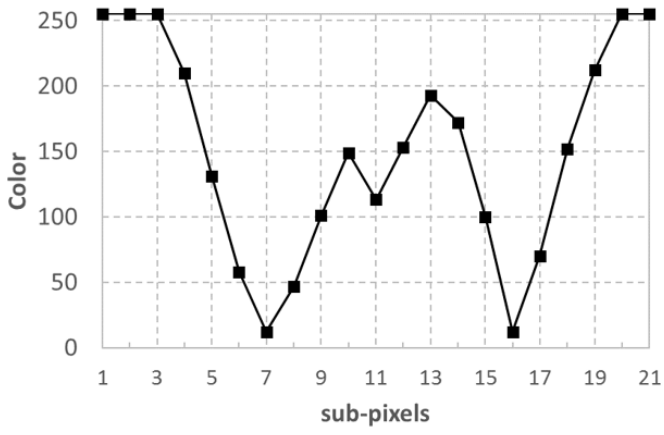
제 4 장 Local index 코딩

본 장에서는 local index 코딩 방식에 대해서 설명한다. Local index 코딩은 텍스트 블록 내에서 각각의 row마다 나타나는 gradient 특징을 반영한 코딩 방법이다. 3장에서 제시한 global index 코딩 방식은 시작 컬러 값과 끝 컬러 값이 '0' 이나 '255' 인 global index 코딩 후보 중에 가장 빈번하게 발생하는 gradient에 대하여 적은 bit로 코딩할 수 있었다. 하지만 global index 방식을 사용할 수 없는 gradient에 대해서는 SPGC 방식을 사용하여야 한다. 예를 들어 그림(4.1) (a)의 ② gradient의 경우 끝나는 지점의 컬러 값이 '255' 로 증가하는 방향의 gradient이기 때문에 끝 지점의 컬러 값이 '0이' 나 '255' 로 끝난다는 것을 나타내는 index 1bit와 gradient를 구성하는 sub-pixel의 개수를 표현하기 위한 비트로 코딩이 가능하다. 하지만 그림(4.1)의 (a)에서 ① gradient의 경우 gradient가 끝나는 지점의 컬러 값이 '0' 이나 '255' 가 아니기 때문에 추가적으로 8bit를 사용하여 컬러 값을 코딩해야 한다. 또한 그림(4.1) (b)와 같이 끝나는 지점의 컬러 값이 '0' 이나 '255' 가 아닌 gradient가 연속적으로 나오는 경우에도 각각의 gradient에 대하여 gradient가 끝나는 지점의 컬러 값을 모두 코딩해야 한다. 이와 같이 sub-pixel gradient 코딩을 사용해야 하는 gradient 중에서도 끝 값을 직접 코딩해야 하는 gradient는 컬러 값 8bit를 추가적으로 비트스트림에 저장해서 압축해야 하므로 압축률을 크게 감소시키는 원인이 된다.



\

(a)



(b)

그림 4.1 텍스트 블록에 존재하는 gradient 형태

따라서 본 장에서는 이러한 압축률 저하를 막기위해 텍스트 블록에서 low 단위로 나타나는 gradient의 특징을 활용한 local 코딩 방식을 제안한다.

4.1 Row 단위로 나타나는 gradient 특징

텍스트 블록에서 나타나는 gradient의 경우 row 단위로 특정한 형태를 갖는다. 그림(4.2)은 텍스트 블록에서 몇개의 라인들의 sub-pixel 컬러 값을 그래프로 나타낸 것이다. 그래프에 빨간 박스로 표시된 부분들은 각각의 라인에서 반복적으로 나타나는 gradient의 형태를 표시한 것이다.

그림에서 확인할 수 있는 것처럼 각각의 row에서 비슷한 형태의 gradient가 반복적으로 나타난다. 이는 텍스트 블록에서 같은 문자가 존재할 경우 sub-pixel 컬러 값 또한 같은 형태로 나타나기 때문이다. 또한 서로 다른 문자라도 같은 라인에 위치한 sub-pixel 컬러 값들이 유사한 형태를 갖는 경우가 발생한다.

그림(4.3)은 서로 다른 'a', 'c', 'o' 3개의 문자에 대해 de-colorization 변환 과정을 거친 후 sub-pixel 단위로 나타낸 것이다. 그림(4.4)의 (a) 와 (b)는 각각 그림(4.3)에서 빨간 박스로 표시한 첫 번째 줄과 다섯 번째 줄의 sub-pixel 컬러 값들을 그래프로 나타낸 것이다. 서로 다른 문자임에도 불구하고 3개의 문자가 서로 유사한 sub-pixel 컬러를 갖는 gradient의 형태를 띄는 것을 확인할 수 있다.

이처럼 텍스트 블록에서 각각의 row는 다양한 이유로 반복적인 형태의 sub-pixel 컬러 값을 갖는다. 본 논문에서는 이와 같이 여러 개의 gradient가 모여서 반복적인 형태를 나타내는 gradient의 묶음을 패턴이라 정의한다.

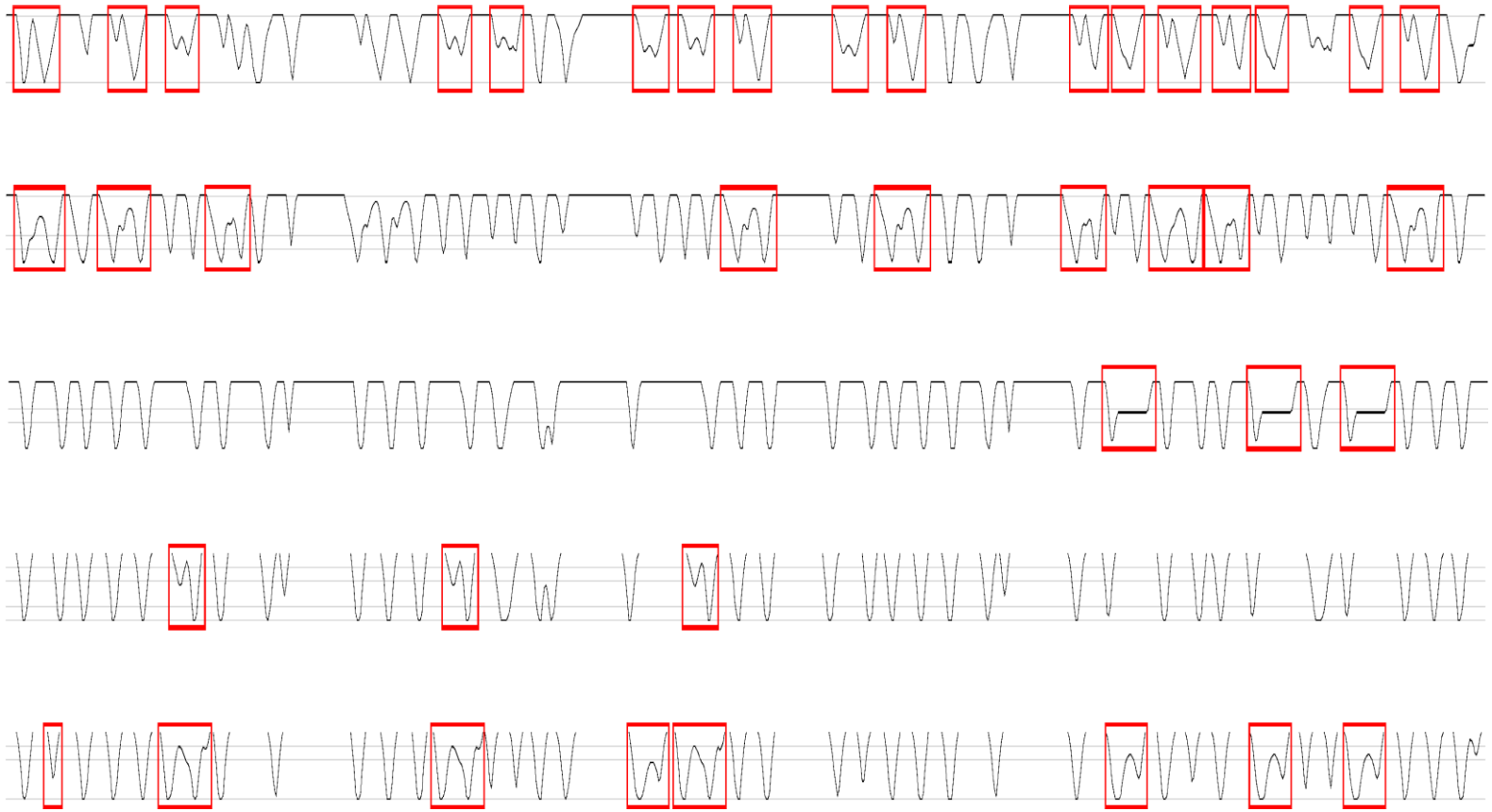
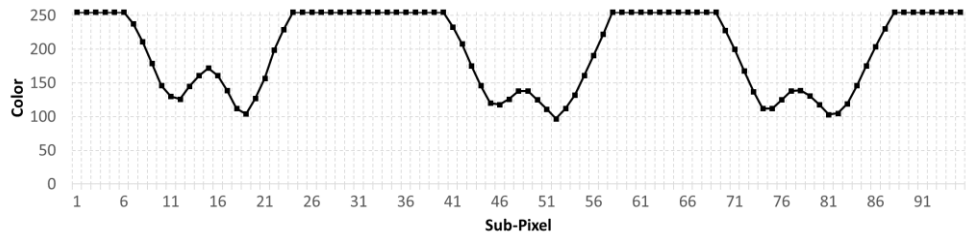
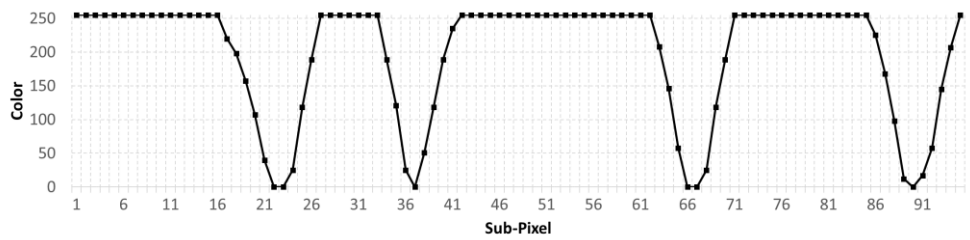


그림 4.2 Row 단위로 나타나는 gradient의 반복적인 형태

그림 4.3 De-colorization 변환 과정을 거친 text ‘a, c, o’



(a)



(b)

그림 4.4 [그림 4.3]에 표시된 row들의 sub-pixel 컬러 분포

4.2 Local index 코딩 방법

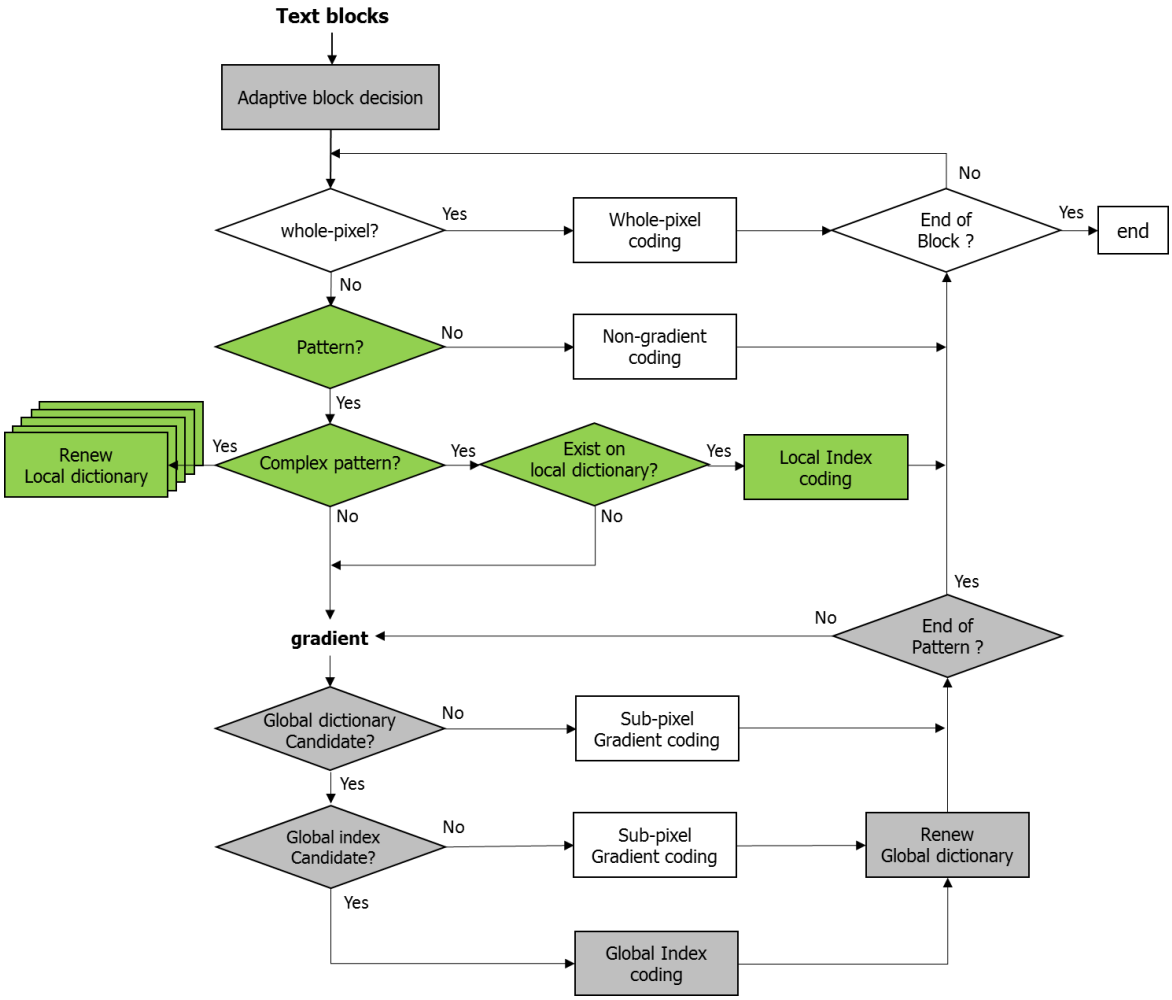


그림 4.5 Local index 코딩 방법

Local index 코딩 방식은 앞에서 살펴본 것과 같이 row 단위로 나타나는 gradient의 패턴을 dictionary에 저장하고 해당 row에서 같은 패턴이 다시 나타나면 패턴을 이루는 여러 개의 gradient를

dictionary의 index로 코딩하는 방식이다.

Local index 방식의 전체적인 흐름도는 그림(4.5)와 같다. 앞에서 설명한 global index 방식에서 추가된 부분을 녹색으로 음영 처리 하였다. Gradient를 압축하기 전에 패턴 단위로 local index 코딩 후보인지 확인하여 후보가 아니면 패턴을 gradient로 나눠서 앞에서 설명한 global index 방식으로 압축한다. 만약 local index 코딩 후보라면 local index 코딩 방식으로 압축하고 local dictionary를 업데이트 한다. 다음은 이러한 local index 코딩 방법에 대해서 자세히 설명 하도록 한다.

4.2.1 패턴의 분류 및 local index 코딩 후보

패턴은 sub-pixel의 컬러 값이 '255' 에서 다시 '255' 가 될 때까지의 나타나는 sub-pixel의 집합으로 정의한다. 그림(4.7)은 그림(4.6)의 문자 't' 에서 1, 2, 3 의 라인에 해당하는 sub-pixel의 컬러 값을 나타낸 것이며 모두 '255' 에서 시작해서 다시 '255' 로 끝나는 패턴을 하나씩 가지고 있다. 패턴은 대표적으로 그림(4.7)에 나타나 있는 것처럼 3가지가 존재한다. 그림(4.7)의 두 번째 패턴의 경우 패턴을 구성하는 gradient 모두 global index 코딩 후보이다. 따라서 경우에 따라 global index 방식이나 SPGC 방식으로 압축된다. 그림(4.7)의 첫 번째 패턴 같은 경우, 패턴을 구성하는 2개의 gradient가 모두 global index 코딩 후보가 아니므로 SPGC 방식을 이용해서 압축해야 한다. 앞부분 '150' 으로 감소하는 gradient는 sub-pixel의 개수와 gradient가 끝나는 지점의 컬러 값 '150' 을 저장해야 하며, 뒤의 '255' 로 증가하는 gradient의 경우는 끝나는

지점의 컬러 값이 '255' 이므로 sub-pixel의 컬러 값과 index 1bit를 저장해야 한다. 그림(4.7)의 세 번째 패턴의 경우는 첫 번째 패턴과 마찬가지로 모든 gradient를 SPGC 방식으로 압축해야 하며, 저장해야 하는 gradient의 끝 지점의 컬러 값이 더욱더 많아진다. 이를 통해 첫 번째와 세 번째 패턴이 global index 코딩 방식의 압축률 저하에 큰 영향을 미친다는 것을 알 수 있다.

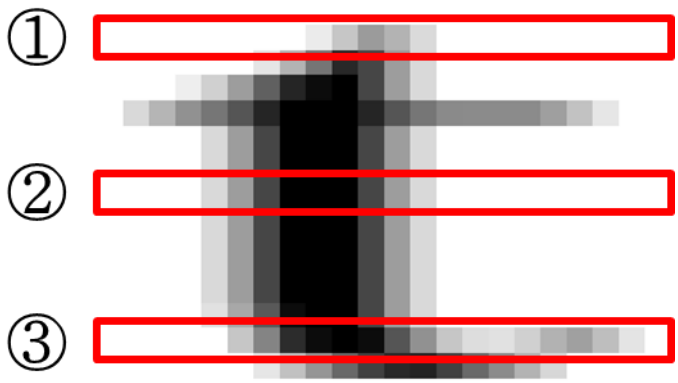


그림 4.6 De-colorization 변환 과정을 거친 text 't'

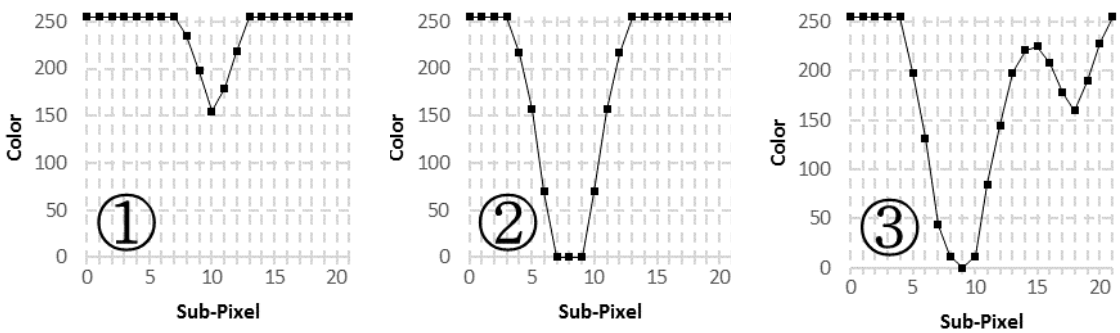


그림 4.7 [그림 4.6]에 표시된 row들의 sub-pixel 컬러 분포

따라서 본 논문에서는 그림(4.7)의 두 번째 패턴과 같이 2개의 global index 코딩 후보 gradient로 이루어진 패턴을 simple 패턴으로 정의하고, 그림(4.7)의 첫 번째와 세 번째 패턴과 같이 gradient의 개수가 2개를 넘거나 global index 코딩 후보가 아닌 gradient가 포함되어 있는 패턴을 complex 패턴이라 정의한다.

Simple 패턴의 경우 텍스트 영역에서 많이 발생하는 패턴이며 global index 방식으로 적은 bit로도 코딩이 가능하다. 하지만 complex 패턴의 경우 global index 방식으로 코딩이 불가능하고, SPGC 방식을 사용하여도 gradient 끝 값을 저장해야 하기 때문에 많은 bit를 필요로 한다. 따라서 local index 코딩 방식에서는 global index 방식으로 코딩이 불가능한 complex 패턴들을 local index 코딩 후보 이용한다.

4.2.2 Local index dictionary vs Global index dictionary

Local index 방식은 global index 방식과 dictionary를 사용하는 방식은 같지만 많은 부분에서 차이가 있다.

첫째로 global index 방식은 텍스트 블록 전 영역에서 발생하는 gradient에 대해 하나의 dictionary를 사용하였다. 하지만 row에서 나타나는 gradient의 패턴은 서로 다른 row에서 서로 다른 패턴들이 나타나기 때문에 텍스트 블록의 각각의 row마다 자신의 dictionary를 필요로 한다. 따라서 row 개수만큼의 dictionary를 생성한다.

둘째로 위와 같이 각각의 row 개수만큼의 dictionary를 생성할 때 global index 코딩 방식에서 사용하는 semi-adaptive dictionary 방식을 사용하면 텍스트 블록의 row 개수만큼의 dictionary를 만들어서 비트스트림에 저장해야 한다. 이는 압축률에 크게 영향을 미칠 수 있다.

따라서 local index 방식에서는 adaptive dictionary 방식을 사용한다. Adaptive dictionary 방식은 encoding/decoding 과정에서 같은 dictionary를 직접 만들어서 사용하기 때문에 semi-adaptive dictionary 방식과 같이 dictionary를 비트스트림에 저장하지 않아도 된다.

셋째로 global index 방식에서는 dictionary에 저장될 global index 코딩 후보가 gradient의 시작과 끝이 '0' 과 '255' 인 gradient를 대상으로 sub-pixel의 개수에 따라 저장하였기 때문에 entry의 개수가 10개를 넘지 않았다. 그렇기 때문에 dictionary의 크기에 크게 영향을 받지 않았다. 하지만 local index 방식의 경우 각각의 row에서 gradient 패턴이 몇 종류가 나타날지 사전에 알 수 없다. 그러므로 dictionary의 크기를 제한하고 dictionary에 entry가 모두 채워졌을 때 기존의 entry를 제거하고 새로운 entry를 추가하기 위한 dictionary 규칙을 필요로 한다.

이와 같은 global index에서 사용하는 global dictionary와의 차이를 고려하여 local index 방식의 local dictionary를 생성한다.

4.2.3 Local index dictionary 생성 및 코딩 방법

Dictionary는 앞에서 설명한 것처럼 각각의 row에서 dictionary를 만들어 저장할 수 없기 때문에 adaptive dictionary 방식으로 dictionary를 생성 및 참조 한다. 기본적으로 row를 코딩하는 과정에서 나타나는 패턴 중에 local index 후보에 해당하는 complex 패턴은 모두 local dictionary에 추가된다. 이때 dictionary에는 complex 패턴을 구성하는 sub-pixel의 컬러 값들이 모두 저장된다. Dictionary는

최종적으로 비트스트림에 저장하지 않고 encoding/decoding 과정에서 각각 동일하게 생성하기 때문에 entry들의 sub-pixel 컬러 값을 모두 dictionary에 저장하여도 압축률에 아무런 영향을 미치지 않는다.

코딩하려는 complex 패턴이 local dictionary에 entry로 이미 저장되어 있다면 해당 entry의 번지수를 index로 사용하여 패턴을 코딩한다. 이때 코딩하려는 complex 패턴과 dictionary에 있는 complex 패턴이 서로 같은 형태의 문자에서 만들어진 패턴이라면 sub-pixel의 컬러 값이 서로 완전히 일치하기 때문에 확인하는데 어려움이 없다. 하지만 그림(4.3)의 서로 다른 3개의 문자에서 만들어지는 패턴은 완전히 일치하지는 않는다. 하지만 서로 상당히 유사하기 때문에 3개를 같은 패턴으로 판단하여 코딩하여도 텍스트의 화질에 거의 영향을 미치지 않는다. 따라서 이렇듯 완전히 일치하지는 않지만 유사한 패턴에 대해서도 local index 코딩을 수행한다. 식(4-1)은 dictionary에 저장되어 있는 패턴과 코딩하려는 패턴의 비교 조건을 나타낸다.

$$\left.\begin{aligned} & N_{d\_sub} - 2 \le N_{c\_sub} \le N_{d_{sub}} + 2 \end{aligned}\right\} \text{Pattern size 조건}$$

$$\left.\begin{aligned} & Sub_i = \left|Color_{d\_i} - Color_{c\_i}\right| < 30 \\ & \sum_{i=1}^{N_{d\_sub}} Sub_i < N_{d\_sub} * 15 \end{aligned}\right\} \text{Sub-pixel value 조건} \qquad (4.1)$$

N_{d_sub}은 dictionary에 있는 패턴을 구성하는 sub-pixel의 개수 이며,

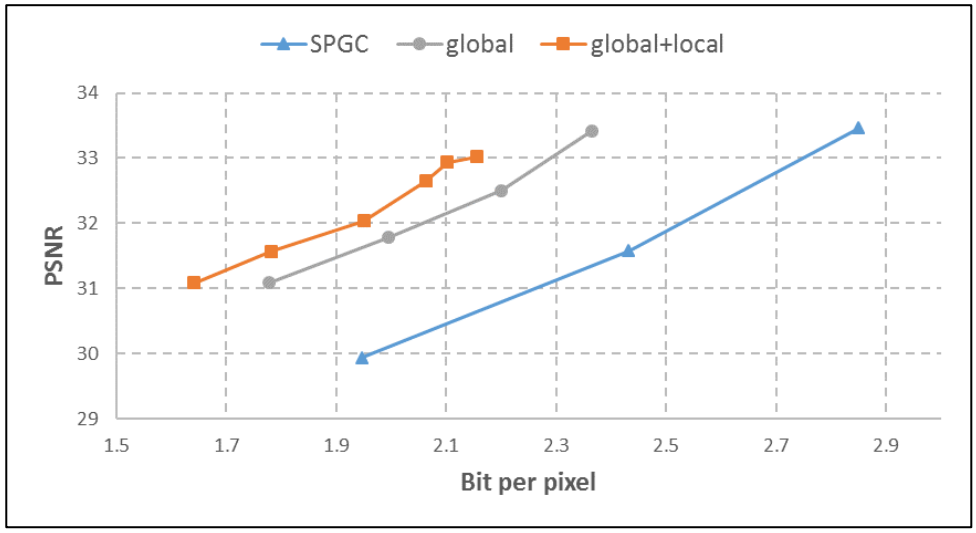
N_{c_sub}은 코딩하려는 패턴을 구성하는 sub-pixel의 개수이다. $Color_{d_i}$는 dictionary에 있는 패턴의 i 번째 sub-pixel 컬러 값 이며, $Color_{c_i}$는 코딩하려는 패턴의 i 번째 sub-pixel 컬러 값 이다. 따라서 첫 번째 조건은 코딩하려는 패턴을 구성하는 sub-pixel의 개수는 dictionary에 저장되어 있는 패턴을 구성하는 sub-pixel의 개수에 대한 조건을 나타내며 두 번째 조건은 각각의 패턴을 구성하는 sub-pixel들의 컬러 값에 대한 조건을 나타낸다. 두 조건을 모두 만족할 경우 서로 같은 패턴으로 보고 dictionary의 index로 패턴을 코딩하는 local index 코딩을 진행한다.

만약 dictionary에 모든 entry가 차서 공간이 부족할 경우에는 LRU(Least Recently Used) 방식을 적용하여 가장 오래 사용되지 않은 entry와 교체하도록 한다.

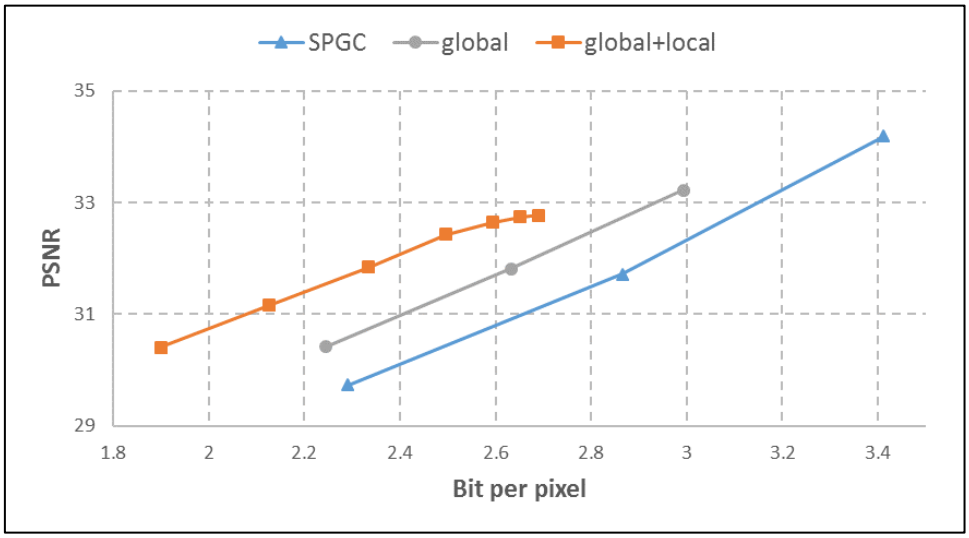
4.3 Local index 코딩 실험 결과

본 실험결과는 local index 코딩 방법의 효율성에 대해 확인한다. 실험은 3장에서 진행한 그림(3.15)의 (b), (c) complex rendered 텍스트 이미지에 대해 진행하며, global index와 local index를 함께 사용한 global + local 방식에 대해 추가적으로 실험을 진행한다. SPGC, global index, global + local index 총 3가지 방식을 비교하도록 한다.

또한 global index 방식과 global + local index 방식의 압축률을 비교하도록 한다.



(a) Complex rendered 텍스트 (1)



(b) Complex rendered 텍스트 (2)

그림 4.8 성능 비교 실험 결과

3장의 global index 방식을 local index 방식과 함께 적용하여 최종 압축 효율과 영상의 화질에 대한 비교는 그림(4.8)에 나타내었다. 그림(4.8)의 (a)는 그림(3.15)(b)의 폰트 크기가 큰 complex rendered 텍스트 이미지에 대한 결과를 나타내며, 그림(4.8)의 (b)는 그림(3.15)(c)의 폰트 크기가 작은 complex rendered 텍스트 이미지에 대한 결과를 나타낸다. Global + local index 방식은 기존의 SPGC 방식보다 압축 효율 측면에서 약 20~25% 정도의 성능 향상을 보였다. 또한 global + local index 방식은 기존의 global index 방식만 사용할 경우보다 PSNR 측면에서 0.5~1 dB 정도 성능 향상을 보여주고 있다. Global index 방식에서는 폰트의 크기가 큰 경우에 global index 후보인 gradient가 많아 global index로 코딩 될 수 있는 gradient가 폰트의 크기가 작을 때 보다 많았다. 따라서 압축 효율 또한 그림(4.8)의 (a)처럼 폰트의 크기가 클 때가 그림(4.8)의 (b)처럼 폰트의 크기가 작을 때 보다 높게 나타났다. 하지만 local index 방식은 global index로 코딩 되지 않는 gradient가 형성하는 complex 패턴에 대해 적용한다. 따라서 global + local index 방식에서는 그림(4.8)의 (a)처럼 폰트의 크기가 클 때 보다 그림(4.8)의 (b)처럼 폰트의 크기가 작을 때 global index 방식만 사용하였을 때보다 압축 효율이 크게 증가한 것을 확인할 수 있다.

다음은 동일한 PSNR에서 global index방법과 비교해서 global + local index 방식의 compression ratio 증가에 대해 확인한다. 표(4.1)은 그림(3.15)(b)의 폰트 크기가 큰 complex rendered 텍스트 이미지에 대한 결과이며, 표(4.2)는 그림(3.15)(c)의 폰트 크기가 작은 complex rendered 텍스트 이미지에 대한 결과이다. 폰트의 크기에 따라 차이는

있지만, global index 방식보다 global + local index 방식을 사용하였을 때 약 10~20% 정도의 압축률을 증가를 확인할 수 있었다.

표 4.1 [그림(3.15)]의 complex rendered 텍스트 (1)

	Global index	Global + Local
PSNR	31.09	31.09
Compression Ratio	13.5	14.93(10.6% ↑)

표 4.2 [그림(3.15)]의 complex rendered 텍스트 (2)

	Global index	Global + Local
PSNR	30.42	30.42
Compression Ratio	10.69	12.63(18.1% ↑)

제 5 장 결 론

본 논문에서는 compound image의 텍스트 블록을 효율적으로 압축하기 위한 알고리즘에 대해 제안하였다. Compound image는 가상화 시스템에서 실시간으로 전송될 필요가 있기 때문에 알고리즘의 압축 효율과 실시간 압축을 필요로 한다. 이러한 조건을 충족시키기 위해 기존의 SPGC 알고리즘은 sub-pixel 영역에서 gradient를 실시간으로 압축함으로써 기존의 텍스트 블록 알고리즘들이나 H.264, JPEG와 같이 널리 사용되는 다른 알고리즘들과 비교해서 높은 압축 효율을 보여주었다. 하지만 텍스트 블록에서 나타나는 gradient의 반복적인 특징을 제대로 활용하지 못하였다. 따라서 본 논문에서 gradient의 텍스트 블록 전체에서 나타나는 반복적인 특징을 활용하여 global index 방식을 제안하였으며, 텍스트 블록의 row 단위로 나타나는 패턴의 반복적인 특징을 활용하여 local index 방식을 제안하였다. 두 가지 알고리즘 모두 dictionary를 이용한 index 코딩 방식을 사용하였다. Global index 방식은 텍스트 블록에서 가장 빈번하게 나타날 가능성이 있는 gradient 만을 dictionary에 저장하여 index 코딩의 효율을 높였으며, local index 방식은 global index 방식으로 코딩 되지 않는 gradient가 이루는 complex 패턴을 dictionary에 저장하여 index 코딩의 효율을 높일 수 있었다.

두 가지 알고리즘을 통해 기존의 SPGC 방식보다 텍스트 영상을 압축하는데 높은 압축 효율과 화질의 향상을 확인할 수 있었다.

제안하는 global index와 local index 알고리즘은 gradient의

반복적인 특징을 활용하여 SPGC방식 보다 20~25% 높은 압축 효율을 보였으며, PSNR 측면에서는 1~1.5dB의 성능 향상을 확인할 수 있었다.

참고 문헌

[1] “Mixed raster content (MRC),” ITU-T Study Group 8, Draft Recommendation T.44, May. 1997

[2] R.de Queiroz, R. Buckley and M. Xu, “Mixed raster context (MRC) model for compound image compression,” in Proc. Of Visual Communications and Image Processing ’99, vol. 3653, pp. 1106-1117, Dec. 1999.

[3] G. Feng and C. A. Bouman. “High-Quality MRC document coding,” IEEE Trans. on Image Processing, vol. 15, no. 10, pp. 3152-3169, Oct. 2006.

[4] R.L. de Queiroz “pre-processing for MRC layers of scanned Images,” 2006 IEEE International Conference on Image Processing, pp. 3093-3096, Oct. 2006.

[5] R. L. de Queiroz, “on data filling algorithms for MRC layers,” in Proc. of 2000 International Conference on Image Processing, Vol. 2. Pp. 586-589, Sep. 2000.

[6] D. Mukherjee, C. Chrysafis and A. Said, “JPEG2000-matched MRC compression of compound documents,” in Proc. of 2002 International Conference on Image Processing, vol. 3, pp. 73-76, 2002.

[7] T. Lin and P. Hao, “Compound image compression for real-time computer screen image transmission,” IEEE Trans. on Image Processing, vol. 14, no. 8, pp. 993-1005, Aug. 2005.

[8] C. Lan, G. Shi and F. Wu, “Compress Compound Image in H. 264/MPGE-4 AVC by Exploiting Spatial Correlation,” IEEE Trans. on Image Processing, vol. 19, no. 4, pp. 946-957. Apr. 2010.

[9] Z. Pan, H. Shen, Y. Lu, S. Li and N. Yu, “A Low-Complexity Screen Compression Scheme for Interactive Screes Sharing,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 23, no. 6, pp. 949-960, Jun. 2013.

[10] Kyudong Kim “Compound image compression using sub-pixel gradient”, Ph.D. dissertation, Dept. ECE., Seoul National Univ., Seoul, Korea, 2014.

[11] Jacob Ziv and Abraham Lempel, “A universal algorithm for sequential data compression,” IEEE Transactions on Information Theory, vol. IT- 23, no. 3, pp. 337–343, May 1977.

[12] T. Porter and T. Duff, “Compositing digital images,” ACM Siggraph Computer Graphics, vol. 18, no. 3, pp. 253-259, Jul. 1984

Abstract

Text block compression on a compound image using a sub-pixel index

Chanhee Park
Department of Electrical and Computer Engineering
The Graduate School
Seoul National University

In this paper, we propose an algorithm to compress text blocks of compound image.

The SPGC algorithm, which encodes the linear features of the sub-pixels in the text block as a gradient, fails to capture the repetitive features of the gradient.

Therefore, in this paper, we propose a method to efficiently compress the gradient considering the repetitive features of the gradient. The proposed method consists of two algorithms: Global Index Compression and Local Index Compression. Both algorithms use the dictionary index method, which stores repeatedly appearing gradients or patterns in a dictionary and codes them into the dictionary's index.

Global Index Compression stores gradients that may appear repeatedly in a text block. The Local Index Compression stores complex patterns of gradients that are not coded by the Global Index Compression method. The maximum index bits of the

dictionary are adaptively determined according to the distribution of gradients in each block, and the index is efficiently assigned by the variable length code method.

Experiments using the proposed algorithm show 20 ~ 25% higher compression efficiency and 1 ~ 1.5dB improvement in PSNR compared with the SPGC method

Keywords : compound image, compression, text block, dictionary, adaptive index, sub-pixel gradient
Student Number : 2015-22787